

# **H- $\alpha$ Reduction of Mosaic 1.1 Images in IRAF's MSCRED Package**

**Ben Hendrickson and Dr. Adriana Durbala**

*updated by Jesse Watson and Dr. Adriana Durbala in summer 2012*

## Contents:

Introduction.....	4
Working in Linux and IRAF.....	5
Organizing Your Data.....	10
Reduction.....	13
1) xtalk Correction.....	13
2) Bias correction.....	14
A: Examine Your Images.....	14
B: Create a Master Bias with zerocombine.....	15
C: Apply the Bias Correction and xtalk Correction On All Images but Biases.....	16
3) Domeflat correction.....	17
A: Create Master Domeflats with flatcombine.....	17
B: Check Skyflats for Exposure Levels.....	18
C: Change Header in Twilightflats.....	19
D: Subtract Master Domeflats From Everything but Biases and Domeflats.....	20
E: Flagging Saturated Pixels on _xbd Images.....	21
4) Skyflat/Illumination Correction.....	22
A: Generating Mastersflat.....	22
B: Flatten Master Skyflats with imsurfit.....	22
C: BPM (Bad Pixel Mask) Prep Work.....	25
D: Skyflat Correction and BPM Generation for all Images but Biases, dflats, and Skyflats.....	25
5) Create a Cosmic Rays BPM.....	26
A: Set craverage Parameters.....	26
B: Create a script to generate cosmic ray masks.....	27
C: Creating and Running Your crscript.cl.....	27
6) Improving the WCS (World Coordinate System).....	28
A: Create Backup Files.....	28
B: Setting the World Coordinate System .....	28
C: Coordinate Matching.....	29
D: Improve WCS with mscmatch .....	29
E: mscgetcatalog again .....	33
F: Checking Your Match.....	33
7) Applying Bad Pixel Masks and Cosmic Ray Masks with fixpix.....	34
A: Create backup files.....	34
B: Applying the BPM.....	35
C: Create backup files again.....	36
D: Applying the Cosmic Ray mask (crmask).....	36
8) Merge Chips.....	37
9) Removing the Sky Gradient with mscskysub.....	38
10) Improving the WCS further with mscmatch.....	39
A: Run mscgetcatalog again.....	39
B: Run mscmatch again.....	40
11) Create backup files .....	41
12) Determine the Relative Scalings of Different Images.....	41
A: Run mscgetcat Again.....	41
B: Run mscimatch.....	41

13) Generate the Stacked Image with mscstack.....	42
14) Apply the bad pixel mask to the stacked image.....	44
15) Bring Sky to Zero.....	44
16) Scale images.....	45
A: Record Flux with imexam.....	45
B: Use imarith to scale the R to H-alpha.....	46
17) Final Subtraction.....	46
18) Shift Correction (OPTIONAL).....	47
A: measure the shift in x & y with imexam.....	47
B: Shift one image relative to the other.....	47
C: Go back to step 14 and subtract again.....	47
16) Image Rotation and Cleaning.....	47

# Introduction

Firstly, this cookbook assumes the reader has very little experience working with Linux and IRAF. It's a very basic and direct "how-to" guide to ccd reduction. I encourage you to search out as many of these cookbooks as possible, both to fill in any missing gaps and provide alternative steps you may prefer. A few I've found helpful include:

-A User's guide to CCD Reduction in IRAF by Phillip Massey:  
<http://iraf.net/irafdocs/ccduser3/>

-Melissa Jacquart's Cookbook, Glenn Tiede's Cookbook:  
<https://sites.google.com/site/melissajacquart/astronomy/iraf>

-Local Group Survey MOSAIC Reduction Notes:  
<http://www2.lowell.edu/users/massey/lgsurvey/splog2.html>

-NOAO Deep Wide Field Survey MOSAIC Reduction Cookbook(Januzzi's cookbook):  
<http://www.noao.edu/noao/noaodeep/ReductionOpt/frames.html>

-Glenn Tiede Cookbook:  
<http://physics.bgsu.edu/~tiede/WOCS/moscookbook.html>

It's also a good idea to go over the Mosaic1.1 manual with a highlighter.:  
<http://lsstmail.org/kpno/mosaic/manual/>

Thanks to Dr. Adriana Durbala for taking the time to walk me through this entire process and make it as painless as possible.

# Working in Linux and IRAF

The Linux GUI is pretty straight forward and similar to Windows in basic layout. Just play around on the operating system for a while to familiarize yourself. One note, when you open a folder in Linux, it opens that new folder in a new window every time. To get rid of this problem you need to change the settings. Go to Edit -> Preferences -> Behavior -> check "Always open in browser windows".

## Useful Linux Commands

Note: text entered into the terminal has been denoted with yellow highlighting. Pink was used to show specific files moving in the directory, or manipulated in some way.

### cd

Used for moving around folders within Linux and IRAF. Just enter "cd" followed by a space and then the subfolder name. Ex: `ecl> cd KittPeak8to11Apr`. To go back type "cd" followed by two periods, `cd ..`

### ls

Displays the contents of the folder you're currently working in.

Ex:

```
ecl> cd KittPeak8to11Apr
```

```
ecl> ls
```

```
20110408 20110411 M11_1011_bpm_.fits unused
20110410 gain_check M11_1011_bpm.fits wcstrial
```

### pwd

If you're really lost, the command "pwd" will display the entire directory path to the folder you're currently in.

Ex:

```
ecl> pwd
```

```
/home/research/KittPeak8to11Apr
```

## mv

The move command will be very useful when you're organizing your files. It allows you to write a renaming script that will be described in detail later. You can also move a file to another location with mv. Here, an image is moved from the "blah" folder to the "research" folder.

```
Ex:
ec1> pwd
/home/research/Benspractice/blah
ec1> ls
blahlist  n1_AGC9573_ha_01_bdsfk.fits  subsets
blahlist~ n1_AGC9573_ha_02_bdsfk.fits  terrible.fits
logfile   n1_AGC9573_ha_03_bdsfk.fits
ec1> mv n1_AGC9573_ha_01_bdsfk.fits /home/research/
ec1> cd ..
ec1> cd ..
ec1> pwd
/home/research
ec1> ls
alfalfaIdl110201 Halpha      n1_AGC9573_ha_01_bdsfk.fits
Benspractice    IDL_programs  practice
BUDDA_2MASS     KittPeak8to11Apr practice_H_alpha
CIG85           logfile      uparm
flat1com.fits   login.cl
```

## cp

"cp" is useful when you want to test out a process and don't want to risk ruining an image you've spent a lot of time on already. "cp" makes a copy of that file and can rename the copy to distinguish the two.

```
Ex:
ec1> ls
alfalfaIdl110201 Halpha      n1_AGC9573_ha_01_bdsfk.fits
Benspractice    IDL_programs  practice
BUDDA_2MASS     KittPeak8to11Apr practice_H_alpha
CIG85           logfile      uparm
flat1com.fits   login.cl
ec1> cp n1_AGC9573_ha_01_bdsfk.fits test
ec1> ls
alfalfaIdl110201 Halpha      n1_AGC9573_ha_01_bdsfk.fits
Benspractice    IDL_programs  practice
BUDDA_2MASS     KittPeak8to11Apr practice_H_alpha
CIG85           logfile      test
```

## del

The delete command. Pretty straight forward just type “del” followed by the file name.

Ex:

```
ecl> ls
alfalaidl110201 Halpha      n1_AGC9573_ha_01_bdsfk.fits
Benspractice IDL_programs  practice
BUDDA_2MASS KittPeak8to11Apr practice_H_alpha
CIG85      logfile      test
flat1com.fits login.cl      uparm
ecl> del test
ecl> ls
alfalaidl110201 Halpha      n1_AGC9573_ha_01_bdsfk.fits
Benspractice IDL_programs  practice
BUDDA_2MASS KittPeak8to11Apr practice_H_alpha
CIG85      logfile      uparm
flat1com.fits login.cl
```

## df

Tells you how much space is remaining on your harddrive(s). You're likely going to make several copies of fairly large files. Make sure you have the space to keep working

```
ecl> df
Filesystem      1K-blocks    Used   Available Use%   Mounted on
/dev/mapper/VolGroup00-LogVol00
468001888 297389052 146456268 68%   /
/dev/sda1      101086     35717    60150   38%   /boot
tmpfs          2000156      0    2000156  0%   /dev/shm
/dev/sdb1      480719056  202828  456097028 1%   /mnt/usbdrive
```

## Useful IRAF Commands

Some commands are IRAF specific and can only be used once you've logged in to IRAF.

## cl

The login command. “cl” tells Linux to start running IRAF. You must have a login.cl file and use the “cd” command in the terminal to go to whatever folder contains your specific login.cl file. Once the “cl” command is entered you'll notice that the prompt has changed to “ecl>”.

## epar

Followed by any task, “epar” displays the parameters of that task and allows you to change them to meet your needs.

Ex:

```
quadred> epar ccdproc
```

### I R A F

Image Reduction and Analysis Facility

PACKAGE = quadred

TASK = ccdproc

images = @stanlist\_xbd List of Mosaic CCD images to process  
(output = @stanlist\_xbds) List of output processed images  
(bpmasks= ) List of output bad pixel masks  
(ccdtype= ) CCD image type to process  
(noprocs = no) List processing steps only?

(xtalkco= no) Apply crosstalk correction?  
(fixpix = no) Apply bad pixel mask correction?  
(oversca= no) Apply overscan strip correction?  
(trim = no) Trim the image?

...

Press :q “enter” to quit and save the parameters.

Press :go “enter” to save the parameters and run the task.

## lpar

Similar to epar, lpar displays the task parameters but doesn't allow you to change them in any way.

## hedit

Occasionally the image headers (image information contained in each .fits file) needs to be edited in order for IRAF to work properly. hedit allows you to change the parameters and work some kinks out of the reduction process.

The IRAF package used to reduce mosaic images is mscred. The following commands are part of that package. Type “mscred” after logging into IRAF to open.

## mscdisp

Followed by an image file name, “mscdisp” will display an image in ds9. ds9 must be open for it to work. (See more information on ds9 on page 8) You can of course just open an image through ds9, but it won't be displayed in the same way. The “disp” command gives you an alternative way to show your image, and may do a better job of showing dust rings, gradients, etc...



## mscexam

mscexam is another way to display an image. Did a better job of showing flaws in the images than simply opening the image with ds9.

:q to quit.

## mscarith

Very useful in checking IRAF's work. For instance, if you're unsure whether or not IRAF has actually changed your image you can use imarith as follows

```
PACKAGE = mscred
```

```
TASK = mscarith
```

```
operand1= n3_d3_b2062_00r_01_xbdswwffi.fits Operand image or numerical constant
```

```
op = - Operator
```

```
operand2= sky.fits Operand image or numerical constant
```

```
result = resid.fits Resultant image
```

```
(extname= ) Extension names to select
```

```
(title = ) Title for resultant image
```

```
(divzero= 0.) Replacement value for division by zero
```

```
(hparams= ) List of header parameters
```

```
(pixtype= ) Pixel type for resultant image
```

```
(calctyp= ) Calculation data type
```

```
(verbose= no) Print operations?
```

```
(noact = no) Print operations without performing them?
```

```
(fd1 = )
```

```
(fd2 = )
```

```
(fd3 = )
```

```
(mode = ql)
```

Here we were subtracting the background sky from an object frame.

## ds9

In order to view your images you need to open them in a program outside of IRAF called ds9. To open a program outside of IRAF you have to put a "!" before the name to let IRAF know it's not an internal task, and a "&" after to continue using IRAF after you've opened an image. Ex: **ec!> !ds9&**

To open a Mosaic specific image in ds9 use File-> Open Other -> Open Mosaic IRAF

If you'd like to view more than one image at once (for comparisons usually) go to the Frames menu and select "New Frame." Then, go back into that menu and select "Tile Frames." From there you can match the images and even blink between two images. This is very useful for checking your processing steps.

One thing you'll have to do in order to properly view your images is to change the login.cl file to reflect the proper size of the images you're working with. To do so, double-click your login.cl file and **set stdimage = imt2048**. (You may have a different size image) If the login.cl file isn't configured properly ds9 may not actually display the entire image.

# Organizing Your Data

Before you start any of the reduction steps it's usually a good idea to organize your images in a way that reflects their distinctive qualities. Begin by creating folders for every type of image, i.e. galaxy, biases, domeflats...

After the images are in their appropriate folders they'll need to be renamed so you can navigate more easily within IRAF.

We chose to specify:

1. Night image was taken. "n1"
2. Type of exposure. "bias"
3. Type of filter. "000" for biases, "00r" for "r" filter, "0h4" for h-alpha4, "h16" for h-alpha16.
4. Number within series of images. "00,01,02..."
5. Only for galaxy images: add the dither number after the night #.

Ex: n1\_dflats\_00r\_04.fits  
n1\_d1\_b2061\_h16\_01.fits

The sheer volume of files will likely make this task very laborious. In order to save time it would be wise to write a simple script to do the job for you. In each folder right-click and create a new empty file. Right-click on the file and rename as "rename\_script.cl" or in the terminal, type `cl>!gedit rename_script.cl&`. The terminal will automatically create the empty file and name it for you.

Then, enter your information in the format below.

```
mv 20110404.koopmann.001.fits n1_bias_000_01.fits
mv 20110404.koopmann.002.fits n1_bias_000_02.fits
mv 20110404.koopmann.003.fits n1_bias_000_03.fits
mv 20110404.koopmann.004.fits n1_bias_000_04.fits
mv 20110404.koopmann.005.fits n1_bias_000_05.fits
```

.....

Save and close the script. In the terminal, type, "`cl < rename_script.cl`" to execute the script. Finally, type "ls" to confirm that your files have been renamed. Repeat for all folders.

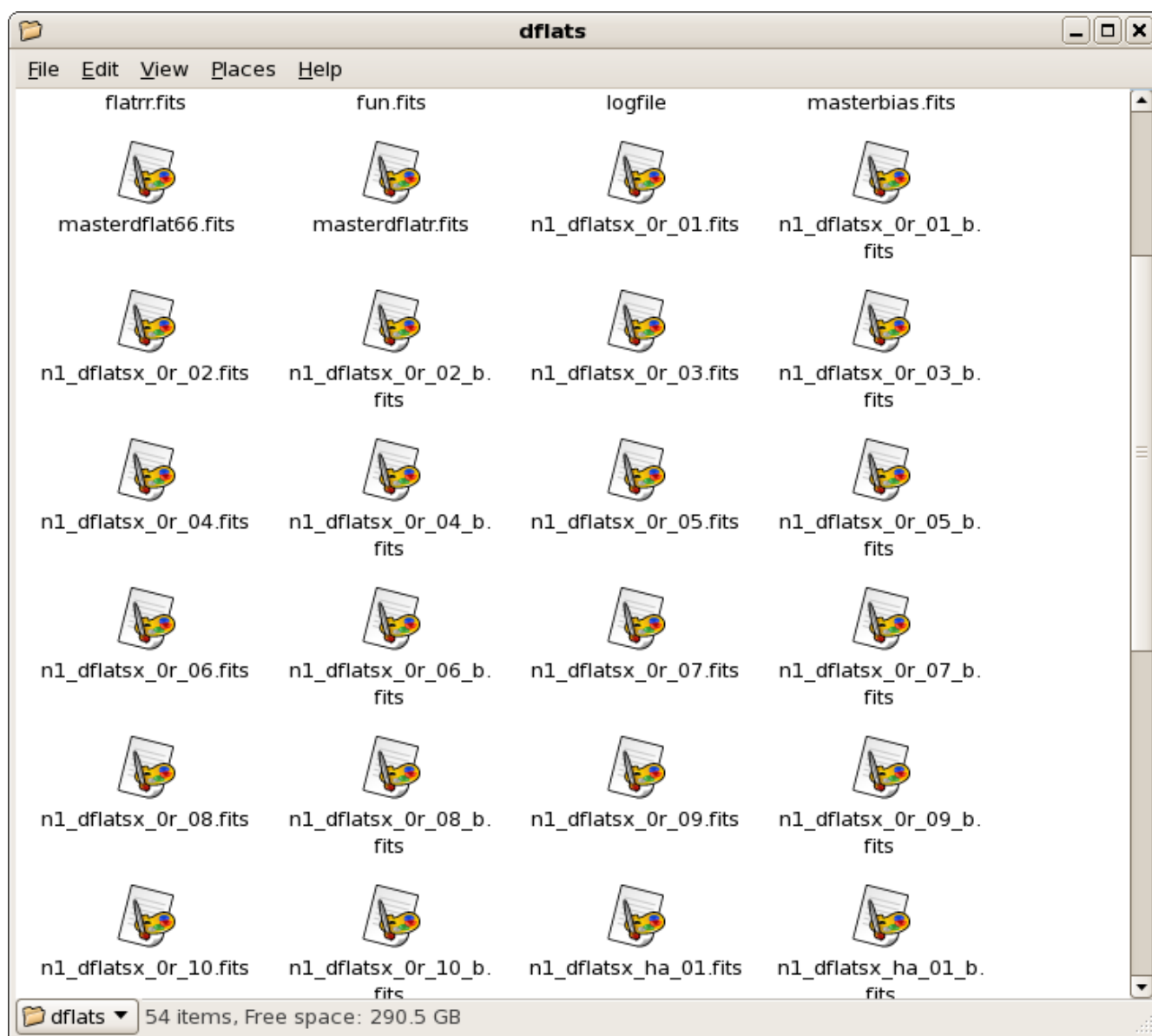


Figure 1: Properly labeled image folder

## Creating Lists

With the files properly named, you can now make image lists for inputs in IRAF. Create a new empty file and name it something specific to a type of image (biaslist, dflatlist, ...). Then, when running tasks in IRAF use the name of your lists as inputs and outputs. The "@" character is used to tell IRAF that input/output is a list and must be put directly in front of the name (@biaslist). Making lists can feel like tedious busy work, but is a great way to ensure that the correct images are being processed at any given point.

**biaslist.list**

Ex: n1\_bias\_000\_01.fits  
 n1\_bias\_000\_02.fits  
 n1\_bias\_000\_03.fits

# Setting up IRAF for Mosaic Reduction

Packages needed to reduce KPNO Mosaic images include: stsdas, tables, mscred, msbdb, fitsutil, and proto.

## 1: Set Up mscred Procedures

Parameter files did not exist for Mosaic on the 36-inch telescope, so I edited existing files for the 4m. These files, Mosaic 1.1.cl, Mosaic1.1.dat, Mosaic1.1.pars are attached. They should be placed in the /iraf/extern/msbdb/noao/kpno/36inch directory. Check that M11\_1011\_xtalk.txt and M11\_1011\_bpm.fits exist in the /iraf/extern/msbdb/noao/Mosaic1.1/ directory. Also check that there are no updates on the Mosaic Calibration page: <http://www.noao.edu/noao/mosaic/calibs.html>

## 2: Run setinstrument

msred

Run **epar setinstrument** and set the parameters as follows:

PACKAGE = mscred  
TASK = setinstrument

```
site =      kpno Site (? for menu)
telescope=  36inch Telescope (? for menu)
instrume=   Mosaic1.1 Instrument (? for a list)
(directo=   msbdb$noao/) Instrument directory
(review =   yes) Review instrument parameters?
query_si=   Site (? for menu or q to quit)
query_te=   Telescope (? for menu or q to quit)
query_in=   Instrument (? for menu or q to quit)
(mode =     ql)
```

:go through first menu & CTRL-D out of every other menu

**ccdlist \*.fits** (do it for “domeflats” folder first, because it contains all the filters)

setinstrument/ccdlist will create the “subsets” file that's necessary for iraf to detect filter type.

```
EX: 'R Harris k1004' R
    'r SDSS k1018' r
    'ha16 H-alpha+16nm k1013' ha16
    'ha8 H-alpha+8nm k1011' ha8
```

You can copy the “subsets” file in every other folder.

# Reduction

Procedure developed via consultation of Buell Jannuzi's Guide to NOAO Deep Wide-Field Survey Mosaic Reductions: <http://www.noao.edu/noao/noaodeep/ReductionOpt/frames.html>

## 1: xtalk Correction for Biases

Crosstalk is a phenomenon that occurs due to the size of the mosaic detector. The NOAO manual has plenty of information about the theory behind it, but suffice it to say, it's thought to be a natural event that occurs due to the large number of amplifiers involved in the imaging process. Bright objects in one amp will cause a dimmer ghost object in another. The mscred package includes a task that corrects this problem. The calibration file can be found on the mosaic website (<http://www.noao.edu/noao/mosaic/kpno/36inch/wcs/monsoon/wcs2011.html>).

Run ccdproc with calibration file on all biases

**PACKAGE = mscred**

**TASK = ccdproc**

**images =** @biaslist List of Mosaic CCD images to process  
**(output =** @biaslist\_x) List of output processed images  
**(bp masks=** ) List of output bad pixel masks  
**(ccdtype=** ) CCD image type to process  
**(noprocs =** no) List processing steps only?

**(xtalkco=** yes) Apply crosstalk correction?  
**(fixpix =** no) Apply bad pixel mask correction?  
**(oversca=** yes) Apply overscan strip correction?  
**(trim =** yes) Trim the image?  
**(zerocor=** no) Apply zero level correction?  
**(darkcor=** no) Apply dark count correction?  
**(flatcor=** no) Apply flat field correction?  
**(sflatco=** no) Apply sky flat field correction?  
**(split =** no) Use split images during processing?  
**(merge =** no) Merge amplifiers from same CCD?

**(xtalkfi=** mscdb\$noao/Mosaic1.1/M11\_1011\_xtalk.txt) Crosstalk file  
**(fixfile=** mscdb\$noao/Mosaic1.1/M11\_1011\_bpm) List of bad pixel masks  
**(saturat=** !saturate) Saturated pixel threshold  
**(sgrow =** 0) Saturated pixel grow radius  
**(bleed =** mean+5000) Bleed pixel threshold  
**(btrail =** 15) Bleed trail minimum length  
**(bgrow =** 1) Bleed pixel grow radius  
**(biassec=** !biassec) Overscan strip image section  
**(trimsec=** !trimsec) Trim data section  
**(zero =** Zero) List of zero level calibration images  
**(dark =** Dark) List of dark count calibration images

```

(flat =      Flat*) List of flat field images
(sflat =    Sflat*) List of secondary flat field images
(minrepl=    1.) Minimum flat field value

(interac=    no) Fit overscan interactively?
(funcio=    legendre) Fitting function
(order =    1) Number of polynomial terms or spline pieces
(sample =    *) Sample points to fit
(naverag=    1) Number of sample points to combine
(niterat=    1) Number of rejection iterations
(low_rej=    3.) Low sigma rejection factor
(high_re=    3.) High sigma rejection factor
(grow =      0.) Rejection growing radius
(fd =        )
(fd2 =        )
(mode =      ql)

```

Reduced counts from 600 to around 0.

## **2: Bias Correction**

### **A: Examine Your Images**

In order to ensure that your bias is comprised of good frames you need to exam them with the mscstat task.

```

PACKAGE = mscrd
TASK = mscstat

```

```

images = n3_00_bias_000_02_x.fits Images
(extname=    ) Extension name selection

```

```

(usemask=    no) Use mask in BPM keyword?
(gmode =    no) Global mode statistics?
(fields =    image, mean) Fields to be printed
(lower =    INDEF) Lower cutoff for pixel values
(upper =    INDEF) Upper cutoff for pixel values
(nclip =    0) Number of clipping iterations
(lsigma =    3.) Lower clipping factor in sigma
(usigma =    3.) Upper clipping factor in sigma
(binwidth=    0.1) Bin width of histogram in sigma
(format =    yes) Format output and print column labels?

(fd1 =      )
(fd2 =      )
(mode =      ql)

```

a) You can make a script "mscstat.cl":

```

mscstat n3_00_bias_000_01_x.fits
mscstat n3_00_bias_000_02_x.fits
.....

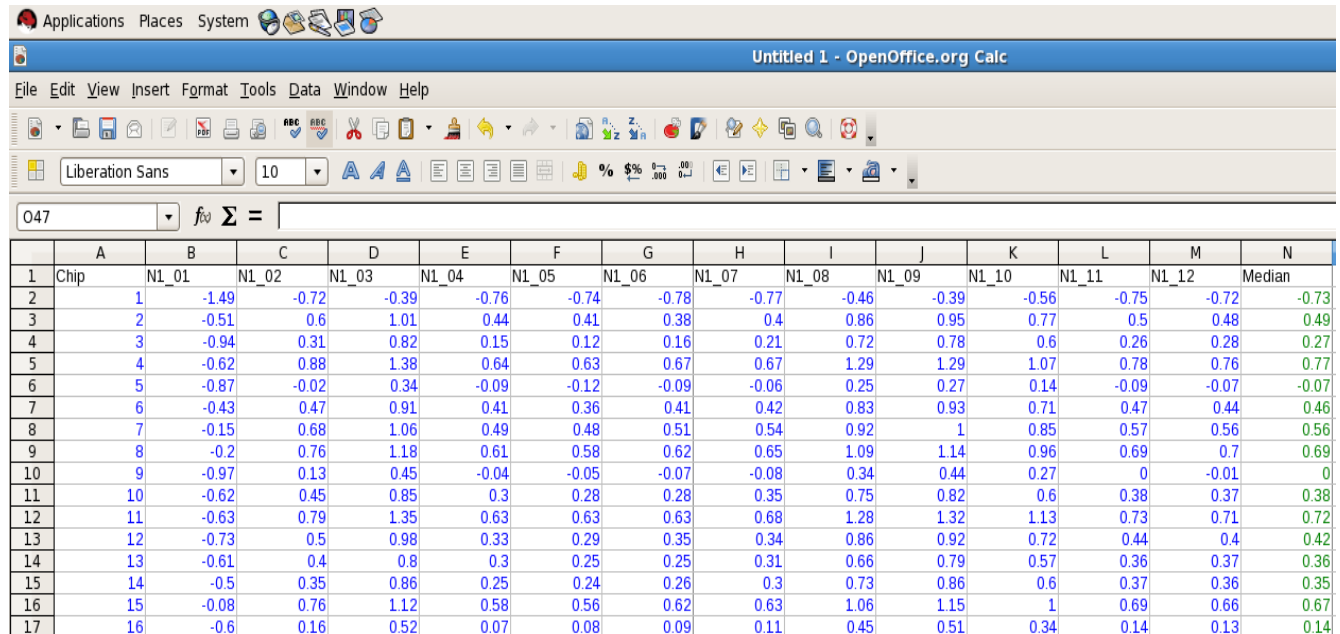
```

run the script: **cl < mscstat.cl > results\_mscstat**

or

- b) Because the 16 chips make everything 16 times harder, you may want to set up an excel spreadsheet in order to look at all the data at once.

The Chips are the rows while the images are the columns. After you run mscstat, with (fields = mean), copy and paste with the middle button into calc. Then, in the column to the right of your data type =median(b?:m?) The question marks should be replaced by the row number.



	A	B	C	D	E	F	G	H	I	J	K	L	M	N
1	Chip	N1_01	N1_02	N1_03	N1_04	N1_05	N1_06	N1_07	N1_08	N1_09	N1_10	N1_11	N1_12	Median
2	1	-1.49	-0.72	-0.39	-0.76	-0.74	-0.78	-0.77	-0.46	-0.39	-0.56	-0.75	-0.72	-0.73
3	2	-0.51	0.6	1.01	0.44	0.41	0.38	0.4	0.86	0.95	0.77	0.5	0.48	0.49
4	3	-0.94	0.31	0.82	0.15	0.12	0.16	0.21	0.72	0.78	0.6	0.26	0.28	0.27
5	4	-0.62	0.88	1.38	0.64	0.63	0.67	0.67	1.29	1.29	1.07	0.78	0.76	0.77
6	5	-0.87	-0.02	0.34	-0.09	-0.12	-0.09	-0.06	0.25	0.27	0.14	-0.09	-0.07	-0.07
7	6	-0.43	0.47	0.91	0.41	0.36	0.41	0.42	0.83	0.93	0.71	0.47	0.44	0.46
8	7	-0.15	0.68	1.06	0.49	0.48	0.51	0.54	0.92	1	0.85	0.57	0.56	0.56
9	8	-0.2	0.76	1.18	0.61	0.58	0.62	0.65	1.09	1.14	0.96	0.69	0.7	0.69
10	9	-0.97	0.13	0.45	-0.04	-0.05	-0.07	-0.08	0.34	0.44	0.27	0	-0.01	0
11	10	-0.62	0.45	0.85	0.3	0.28	0.28	0.35	0.75	0.82	0.6	0.38	0.37	0.38
12	11	-0.63	0.79	1.35	0.63	0.63	0.63	0.68	1.28	1.32	1.13	0.73	0.71	0.72
13	12	-0.73	0.5	0.98	0.33	0.29	0.35	0.34	0.86	0.92	0.72	0.44	0.4	0.42
14	13	-0.61	0.4	0.8	0.3	0.25	0.25	0.31	0.66	0.79	0.57	0.36	0.37	0.36
15	14	-0.5	0.35	0.86	0.25	0.24	0.26	0.3	0.73	0.86	0.6	0.37	0.36	0.35
16	15	-0.08	0.76	1.12	0.58	0.56	0.62	0.63	1.06	1.15	1	0.69	0.66	0.67
17	16	-0.6	0.16	0.52	0.07	0.08	0.09	0.11	0.45	0.51	0.34	0.14	0.13	0.14

Ran through biases for all three nights, the only one with significant deviation was n3\_01(night 3), which will be discarded.

## B: Create a Master Bias With zerocombine

This task creates the bias image that will be used in correction for all your other images. Our parameters may not necessarily going to be the best option for you. After some tedious experimentation we discovered that the RDNOISE value in our headers was way off. Consequently, we chose to use avsigclip as the rejection parameter, since avsigclip ignores both RDNOISE and GAIN in its routine. A more standard rejection type is ccdclip for smaller sets of images and crreject for large sets. I recommend doing a bit of research to determine which will work best for you.

USE CRREJECT FOR REJECTION IF RDNOISE VALUES HAVE BEEN CORRECTED, OTHERWISE USE AVSIGCLIP.

Run this for every night.

```
PACKAGE = mscred
TASK = zerocombine
```

```
input = @biaslist_x List of zero level images to combine
(output = masterbias) Output zero level name
(combine= median) Type of combine operation
(reject = avsigclip) Type of rejection
(ccdtype= zero) CCD image type to combine
(process= yes) Process images before combining?
(delete = no) Delete input images after combining?
(scale = none) 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= RDNOISE) ccdclip: CCD readout noise (electrons)
(gain = GAIN) ccdclip: CCD gain (electrons/DN)
(snoise = 0.) ccdclip: Sensitivity noise (fraction)
(pclip = -0.5) pclip: Percentile clipping parameter
(blank = 0.) Value if there are no pixels
(mode = ql)
```

Will create “masterbias.fits”

## C: Apply the Bias Correction and xtalk Correction On All Images but Biases

Now that you have your master bias file it's time for the second run through ccdproc. In addition to the bias correction this is also the point to perform xtalk correction on the rest of your images.

```
>epar ccdproc
```

```
PACKAGE = mscred
TASK = ccdproc
```

```
images = @dflatlist List of Mosaic CCD images to process
(output = @dflatlist_xb) List of output processed images
(bpmasks= ) List of output bad pixel masks
(ccdtype= ) CCD image type to process
(noproc = no) List processing steps only?

(xtalkco= yes) Apply crosstalk correction?
(fixpix = no) Apply bad pixel mask correction?
(oversca= yes) Apply overscan strip correction?
(trim = yes) Trim the image?
(zerocon= yes) Apply zero level correction?
(darkcor= no) Apply dark count correction?
(flatcor= no) Apply flat field correction?
(sflatco= no) Apply sky flat field correction?
(split = no) Use split images during processing?
(merge = no) Merge amplifiers from same CCD?
```



```

(xtalkfi= mscdb$noao/Mosaic1.1/M11_1011_xtalk.txt) Crosstalk file
(fixfile= mscdb$noao/Mosaic1.1/M11_1011_bpm) List of bad pixel masks
(saturat= !saturate) Saturated pixel threshold
(sgrow = 0) Saturated pixel grow radius
(bleed = mean+5000) Bleed pixel threshold
(btrail = 15) Bleed trail minimum length
(bgrow = 1) Bleed pixel grow radius
(biassec= !biassec) Overscan strip image section
(trimsec= !trimsec) Trim data section
(zero = masterbias) List of zero level calibration images
(dark = Dark) List of dark count calibration images
(flat = Flat) List of flat field images
(sflat = ) List of secondary flat field images
(minrepl= 1.) Minimum flat field value

(interac= no) Fit overscan interactively?
(funcio= legendre) Fitting function
(order = 1) Number of polynomial terms or spline pieces
(sample = *) Sample points to fit
(naverag= 1) Number of sample points to combine
(niterat= 1) Number of rejection iterations
(low_rej= 3.) Low sigma rejection factor
(high_re= 3.) High sigma rejection factor
(grow = 0.) Rejection growing radius
(fd = )
(fd2 = )
(mode = ql)

```

Comments: Counts didn't go to zero. For skyflats & domeflats the number of counts increased a bit, for all the other images the number of counts decreased.

### 3: Domeflat Correction

#### A: Create Master Domeflats with flatcombine

Virtually the same as zerocombine. Again, we chose to use avsigclip as the rejection routine to avoid a RDNOISE discrepancy. Note: setinstrument should have created a file, called *subsets*, that will enable IRAF to automatically determine the filter used for any given frame. So feel free to use just one input list and one output list. IRAF will process each filter uniquely and spit out a master dome flat for each filter used.

```

PACKAGE = mscred
TASK = flatcombine

```

```

input = @dflatlist_xb List of flat field images to combine
(output = masterdflat) Output flat field root name
(combine= median) Type of combine operation
(reject = avsigclip) Type of rejection
(ccdtype= ) 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?

```

```

(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=        RDNOISE) ccdclip: CCD readout noise (electrons)
(gain =          GAIN) 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 =          q)

```

Note: If RDNOISE and GAIN are ok, use ccdclip in place of avsigclip.

Creates masterdflatha.fits, masterdflatr.fits, and masterdflatR.fits.

## B: Check Skyflats for Exposure Levels

Just like when you checked over your biases, the goal here is to make sure that your master twilight flats aren't created out of frames with large count deviations. Images should be rejected if counts **70,000 < mode < 150,000**. Use mscstat, and set fields = image, mode . The counts may also be listed in your observation logs. I recommend checking the mode with mscstat against the log info for a few frames before relying on that information.

i) epar mscstat

```

PACKAGE = mscrd
TASK = mscstat

```

```

images = n2_tflat_ha8_03.fits Images
(extname=      ) Extension name selection

```

```

(usemask=      no) Use mask in BPM keyword?
(gmode =      no) Global mode statistics?
(fields =      image, mode) Fields to be printed
(lower =      INDEF) Lower cutoff for pixel values
(upper =      INDEF) Upper cutoff for pixel values
(nclip =      0) Number of clipping iterations
(lsigma =      3.) Lower clipping factor in sigma
(usigma =      3.) Upper clipping factor in sigma
(binwidth=     0.1) Bin width of histogram in sigma
(format =      yes) Format output and print column labels?

```

```

(fd1 =        )
(fd2 =        )
(mode =       ql)

```

```
mscred> mscstat n2_tflat_h16_03.fits
```

```
#   MODE
69084.
81097.14
84246.61
73961.51
70448.84
76923.52
.....
```

ii) Create a script “mscstat\_script” to do mscstat for all skyflats simultaneously:

```
mscstat n2_tflat_h16_01.fits
mscstat n2_tflat_h16_02.fits
mscstat n2_tflat_h16_03.fits
.....
```

iii) Run the script:

```
cl < mscstat_script > results_mscstat
```

## C: Change Header in Twilightflats

Before you can perform the domeflat correction to your images, you must alter the headers on your skyflats. The problem is that IRAF can get confused. It thinks that your skyflats are actually domeflats and it will not apply the correction. In order to avoid any confusion you have to alter the headers in your sky flats to read of OBJECT rather than SKYFLAT.

First make a list of all your skyflats from each night, but this time include each amplifier for every image.

```
EX: n3_tflat_Or_01.fits[im1]
n3_tflat_Or_01.fits[im2]
n3_tflat_Or_01.fits[im3]
n3_tflat_Or_01.fits[im4]
n3_tflat_Or_01.fits[im5]
n3_tflat_Or_01.fits[im6]
n3_tflat_Or_01.fits[im7]
n3_tflat_Or_01.fits[im8]
n3_tflat_Or_01.fits[im9]
n3_tflat_Or_01.fits[im10]
n3_tflat_Or_01.fits[im11]
n3_tflat_Or_01.fits[im12]
.....
```

Then, just run hedit with the following parameters.

```
PACKAGE = imutil
```

```
TASK = hedit
```

```
images = @tflatlistchips images to be edited
fields = obstype fields to be edited
value = object value expression
(add = no) add rather than edit fields
(addonly= no) add only if field does not exist
(delete = no) delete rather than edit fields
(verify = no) verify each edit operation
(show = yes) print record of each edit operation
(update = yes) enable updating of the image header
(mode = ql)
```

## D: Subtract Master Domeflats From Everything but Biases and Domeflats

Pretty much the same as bias correction. Create backup lists and run through ccdproc with the following parameters. Again, IRAF can determine which image used which filter, so just put the generic name of your image file with an asterisk instead of the filter letter(s).

PACKAGE = mscrd

TASK = ccdproc

images = @2061list\_xb List of Mosaic CCD images to process (Object list NRGb206-1)

(output = @2061list\_xbd) List of output processed images

(bpmmasks= ) List of output bad pixel masks

(ccdtype= ) CCD image type to process

(noproc = no) List processing steps only?

(xtalkco= no) Apply crosstalk correction?

(fixpix = no) Apply bad pixel mask correction?

(oversca= no) Apply overscan strip correction?

(trim = no) Trim the image?

(zerocor= no) Apply zero level correction?

(darkcor= no) Apply dark count correction?

(flatcor= yes) Apply flat field correction?

(sflatco= no) Apply sky flat field correction?

(split = no) Use split images during processing?

(merge = no) Merge amplifiers from same CCD?

(xtalkfi= mscdb\$noao/Mosaic1.1/M11\_1011\_xtalk.txt) Crosstalk file

(fixfile= mscdb\$noao/Mosaic1.1/M11\_1011\_bpm) List of bad pixel masks

(saturat= !saturate) Saturated pixel threshold

(sgrow = 0) Saturated pixel grow radius

(bleed = mean+5000) Bleed pixel threshold

(btrail = 15) Bleed trail minimum length

(bgrow = 1) Bleed pixel grow radius

(biassec= !biassec) Overscan strip image section

(trimsec= !trimsec) Trim data section

(zero = masterbias) List of zero level calibration images

(dark = Dark) List of dark count calibration images

(flat = masterdflat\*) List of flat field images

(sflat = ) List of secondary flat field images

(minrepl= 1.) Minimum flat field value

(interac= no) Fit overscan interactively?

(functio= legendre) Fitting function

(order = 1) Number of polynomial terms or spline pieces

(sample = \*) Sample points to fit

(naverag= 1) Number of sample points to combine

(niterat= 1) Number of rejection iterations

(low\_rej= 3.) Low sigma rejection factor

(high\_re= 3.) High sigma rejection factor

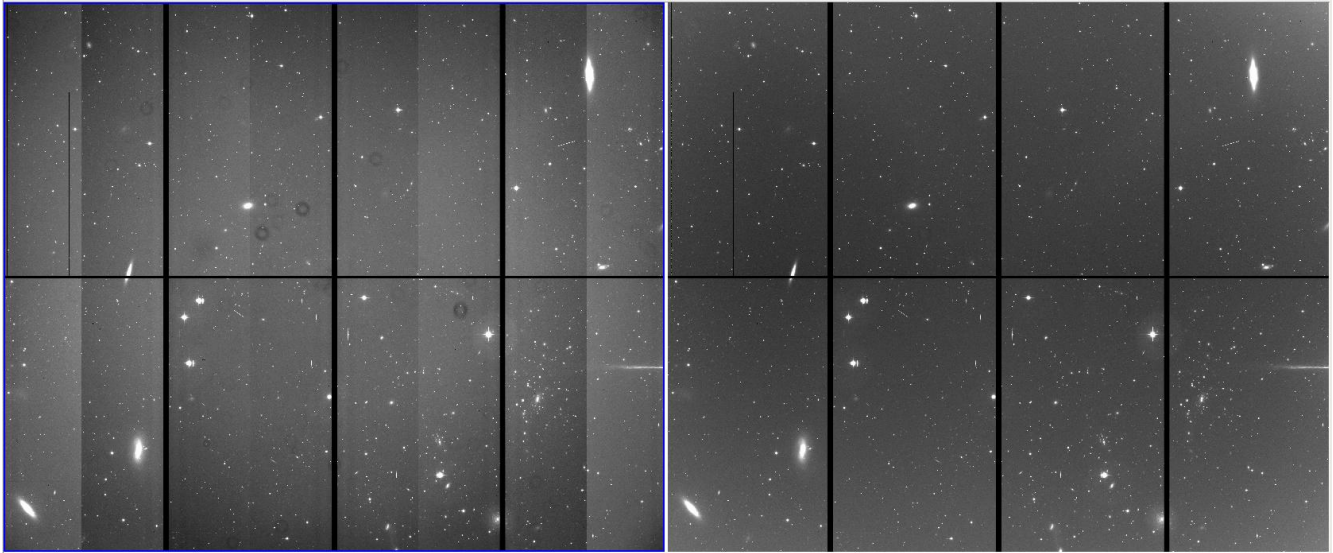
(grow = 0.) Rejection growing radius

(fd = )

(fd2 = )

(mode = ql)

Note: \* = ha, r, R. IRAF will pick up the correct flat for that image.



Transition from raw (no correction) to \_xbd (used msdisp)

## E: Flagging Saturated Pixels on \_xbd Images

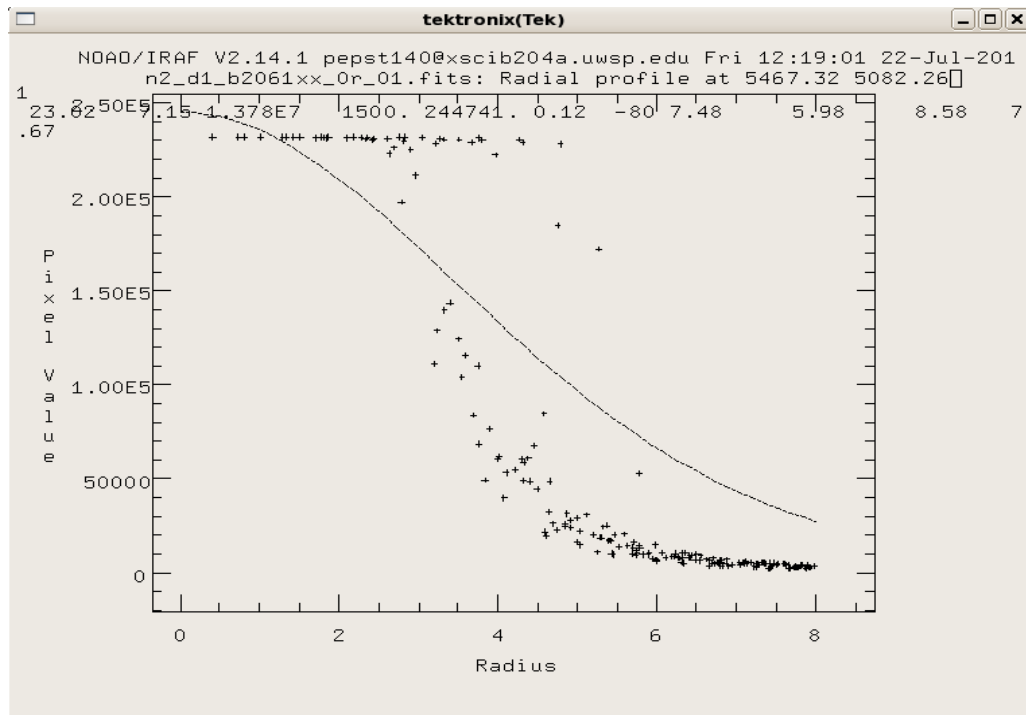
run epar rimexam and set “fit and subtract background” = no. if you don't, the sky will be subtracted everytime you press “r”

Checked each chip with msccexam. Click “r” for a radial plot of the saturation levels.

The saturation was around 200,000. this agreed with the saturation level that was provided in the header.

From header...

SaturateSATURATE= 220000 / Approx saturation (ADU)



## 4: Skyflat/Twilight Correction

### A: Generating Mastersflat

Similar to zerocombine and flatcombine. Run the same way you did before with the following parameters.

```
PACKAGE = mscred
TASK = sflatcombine
```

```
input = @tflatlist_xbd List of images to combine
(output = mastersflat) Output sky flat field root name
(combine= median) Type of combine operation
(reject = avsigclip) Type of rejection
(ccdtype= ) CCD image type to combine
(subsets= yes) Combine images by subset parameter?
(masktyp= none) Mask type
(maskval= 0.) Mask value
(scale = mode) Image scaling
(statsec= ) Image section for computing statistics
(nkeep = 1) Minimum to keep (pos) or maximum to reject (neg)
(nlow = 1) minmax: Number of low pixels to reject
(nhigh = 1) minmax: Number of high pixels to reject
(mclip = yes) Use median in sigma clipping algorithms?
(lsigma = 6.) Lower sigma clipping factor
(hsigma = 3.) Upper sigma clipping factor
(rdnoise= rdnoise) ccdclip: CCD readout noise (electrons)
(gain = gain) 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
(grow = 3.) Radius (pixels) for neighbor rejection
(fd = )
(mode = ql)
```

Note: Use ccdclip instead of avsigclip if RDNOISE and GAIN are ok.

Creates mastersflatha.fits, mastersflatr.fits, and mastersflatR.fits.

### B: Flatten Master Skyflats with imsurfit:

Start by unlearning the parameters of the imfit task imsurfit. Double check by running lpar imsurfit.

```
>unlearn imsurfit
```

```
>epar imsurfit
```

PACKAGE = imfit  
TASK = imsurfit

input = Input images to be fit  
output = Output images  
xorder = 2 Order of function in x  
yorder = 2 Order of function in y  
(type\_ou= fit) Type of output (fit,residual,response,clean)  
(functio= leg) Function to be fit (legendre,chebyshev,spline3)  
(cross\_t= yes) Include cross-terms for polynomials?  
(xmedian= 1) X length of median box  
(ymedian= 1) Y length of median box  
(median\_= 50.) Minimum fraction of pixels in median box  
(lower = 0.) Lower limit for residuals  
(upper = 0.) Upper limit for residuals  
(ngrow = 0) Radius of region growing circle  
(niter = 0) Maximum number of rejection cycles  
(regions= all) Good regions (all,rows,columns,border,sections,c  
(rows = \*) Rows to be fit  
(columns= \*) Columns to be fit  
(border = 50) Width of border to be fit  
(section= ) File name for sections list  
(circle = ) Circle specifications  
(div\_min= INDEF) Division minimum for response output  
(mode = ql)

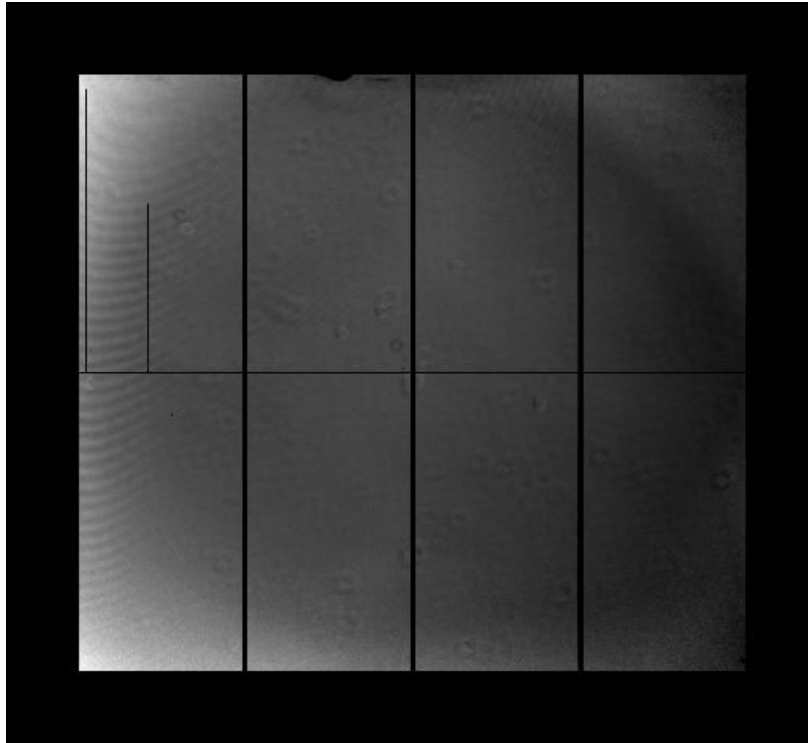
Since imsurfit is not a mscred task we need to use msccmd to run imsurfit.

```
mscred> msccmd  
msccmd: imsurfit $input $output xo=3 yo=3 xm=100 ym=100
```

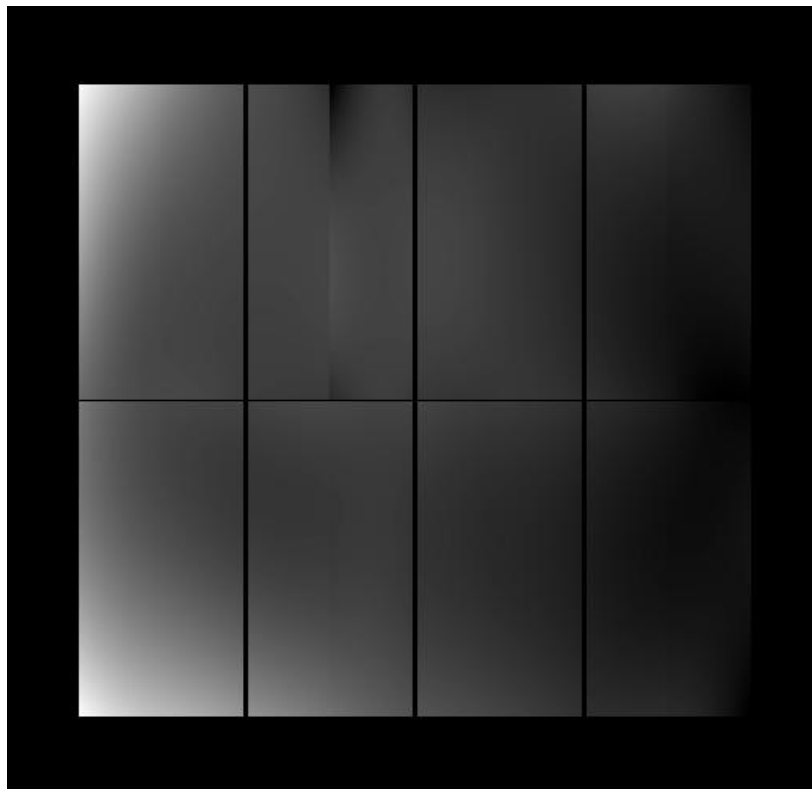
Input files (mastersflatR.fits):  
Output files (mastersflatR\_surf.fits):  
msccmd:

q to get out of msccmd

Now check the work by dividing the new surfaced sky flat over the old master sky flat. The resulting image should have counts around 1. imsurfit will smooth out the image and get rid of the dust rings that were not subtracted well with the domeflat correction.



sky flat before imsurfit (dust rings present)



sky flat after running imsurfit (dust rings corrected)



## C: BPM (Bad Pixel Mask) Prep Work

One last thing remains before applying the final skyflats to the remainder of your images. You want to create bad pixel masks when you apply sky corrections. For this you need to create a list *@masks.list* with the name of your images. that has "bpm\_" added to the beginning. The masks themselves will be created by IRAF and put into individual folders. (Ignore them for now.)

EX:

@masks.list

bpm\_n3\_d5\_b2061\_00r\_01\_xbds    -> in each folder IRAF creates ".pl" files

bpm\_n3\_d5\_b2061\_00R\_01\_xbds

bpm\_n3\_d5\_b2061\_h16\_01\_xbds

Note: The folders are created automatically by IRAF. You only have to create the list.

## D: Skyflat Correction and BPM Generation for all Images but Biases, dflats, and Skyflats

Finally! You're ready to apply your smoothed skyflats to your remaining images. After all that work it's just a standard run through ccdproc. Be sure to check the parameters below.

PACKAGE = mscrd

TASK = ccdproc

images = @2061list\_xbd List of Mosaic CCD images to process

(output = @2061list\_xbds) List of output processed images

(bpmasks= @masks.list ) List of output bad pixel masks

(ccdtype= ) CCD image type to process

(noproc = no) List processing steps only?

(xtalkco= no) Apply crosstalk correction?

(fixpix = no) Apply bad pixel mask correction?

(oversca= no) Apply overscan strip correction?

(trim = no) Trim the image?

(zerocor= no) Apply zero level correction?

(darkcor= no) Apply dark count correction?

(flatcor= no) Apply flat field correction?

(sflatco= yes) Apply sky flat field correction?

(split = no) Use split images during processing?

(merge = no) Merge amplifiers from same CCD?

(xtalkfi= mscdb\$noao/Mosaic1.1/M11\_1011\_xtalk.txt) Crosstalk file

(fixfile= mscdb\$noao/Mosaic1.1/M11\_1011\_bpm) List of bad pixel masks

(saturat= !saturate) Saturated pixel threshold

(sgrow = 0) Saturated pixel grow radius

(bleed = mean+5000) Bleed pixel threshold

(btrail = 15) Bleed trail minimum length

(bgrow = 1) Bleed pixel grow radius

(biassec= !biassec) Overscan strip image section

(trimsec= !trimsec) Trim data section

(zero = masterbias) List of zero level calibration images

(dark = Dark) List of dark count calibration images

```

(flat =      masterdflat*) List of flat field images
(sflat =    mastersflat*_surf.fits)List of secondary flat field images
(minrepl=    1.) Minimum flat field value
(interac=    no) Fit overscan interactively?
(funcio=     legendre) Fitting function
(order =     1) Number of polynomial terms or spline pieces
(sample =    *) Sample points to fit
(naverag=    1) Number of sample points to combine
(niterat=    1) Number of rejection iterations
(low_rej=    3.) Low sigma rejection factor
(high_re=    3.) High sigma rejection factor
(grow =      0.) Rejection growing radius
(fd =        )
(fd2 =       )
(mode =      ql)

```

Note: \* means that IRAF will know to choose the appropriate filter.

## 5: Create a Cosmic Rays BPM

One more mask has to be created to go with the BPM you just created previously.

### A: Set craverage Parameters

Go to package noao.imred.crutil, then run epar craverage with the following parameters. **:q to quit out. Don't actually run the craverage task.**

```

PACKAGE = crutil
TASK = craverage

```

```

input =      List of input images
output =     List of output images
(crmask =    ) List of output cosmic ray and object masks
(average=    ) List of output block average filtered images
(sigma =     ) List of output sigma images

(navg =      7) Block average box size
(nrej =      3) Number of high pixels to reject from the average
(nbkg =      5) Background annulus width
(nsig =      50) Box size for sigma calculation
(var0 =      0.) Variance coefficient for DN^0 term
(var1 =      0.) Variance coefficient for DN^1 term
(var2 =      0.) Variance coefficient for DN^2 term

(crval =     1) Mask value for cosmic rays
(lcrsig =    10.) Low cosmic ray sigma outside object
(hcrsig =    3.75) High cosmic ray sigma outside object
(crgrow =    0.) Cosmic ray grow radius

(objval =    0) Mask value for objects
(lobjsig=    10.) Low object detection sigma
(hobjsig=    5.5) High object detection sigma
(objgrow=    0.) Object grow radius

```

## B: Create a script to generate cosmic ray masks

To actually run the task and create the crmasks, you need create a script (crscript.cl). You could do each chip individually but it would take forever.

Run mscstat to limit the fields to only image.

```
PACKAGE = mscrd
TASK = mscstat
```

```
images =      coco.fits Images
(extname=      ) Extension name selection

(usemask=      no) Use mask in BPM keyword?
(gmode =      no) Global mode statistics?
(fields =      Image) Fields to be printed
(lower =      INDEF) Lower cutoff for pixel values
(upper =      INDEF) Upper cutoff for pixel values
(nclip =      0) Number of clipping iterations
(lsigma =      3.) Lower clipping factor in sigma
(usigma =      3.) Upper clipping factor in sigma
(binwidth=      0.1) Bin width of histogram in sigma
(format =      yes) Format output and print column labels?

(fd1 =      )
(fd2 =      )
(mode =      ql)
```

**mscred> mscstat \*\_xbds.fits > crscript.cl** → IRAF creates automatically “crscript.cl”, don’t need to make an empty file

## C: Creating and Running Your crscript.cl

Start with an image chips list like the one below.

```
#      IMAGE
n2_d1_b2061_00R_01_xbds.fits[im1]
n2_d1_b2061_00R_01_xbds.fits[im2]
n2_d1_b2061_00R_01_xbds.fits[im3]
n2_d1_b2061_00R_01_xbds.fits[im4]
n2_d1_b2061_00R_01_xbds.fits[im5]
n2_d1_b2061_00R_01_xbds.fits[im6]
n2_d1_b2061_00R_01_xbds.fits[im7]
n2_d1_b2061_00R_01_xbds.fits[im8]
n2_d1_b2061_00R_01_xbds.fits[im9]
n2_d1_b2061_00R_01_xbds.fits[im10]
n2_d1_b2061_00R_01_xbds.fits[im11]
n2_d1_b2061_00R_01_xbds.fits[im12]
n2_d1_b2061_00R_01_xbds.fits[im13]
n2_d1_b2061_00R_01_xbds.fits[im14]
n2_d1_b2061_00R_01_xbds.fits[im15]
n2_d1_b2061_00R_01_xbds.fits[im16]
```

```
.....
replace [im1] with [1]
.....
```

The script will ultimately have many decently sized lines. I was able to save a lot of time by doing the major work on Calc/Excel and copy and paste it into gedit.

The final crscript.cl looks like this:

```
craverage n2_d1_b2061_00R_01_xbds.fits[1] output="" crmask="crmask/crmaskn2_d1_b2061_00R_01_1"
craverage n2_d1_b2061_00R_01_xbds.fits[2] output="" crmask="crmask/crmaskn2_d1_b2061_00R_01_2"
craverage n2_d1_b2061_00R_01_xbds.fits[3] output="" crmask="crmask/crmaskn2_d1_b2061_00R_01_3"
craverage n2_d1_b2061_00R_01_xbds.fits[4] output="" crmask="crmask/crmaskn2_d1_b2061_00R_01_4"
craverage n2_d1_b2061_00R_01_xbds.fits[5] output="" crmask="crmask/crmaskn2_d1_b2061_00R_01_5"
craverage n2_d1_b2061_00R_01_xbds.fits[6] output="" crmask="crmask/crmaskn2_d1_b2061_00R_01_6"
craverage n2_d1_b2061_00R_01_xbds.fits[7] output="" crmask="crmask/crmaskn2_d1_b2061_00R_01_7"
craverage n2_d1_b2061_00R_01_xbds.fits[8] output="" crmask="crmask/crmaskn2_d1_b2061_00R_01_8"
craverage n2_d1_b2061_00R_01_xbds.fits[9] output="" crmask="crmask/crmaskn2_d1_b2061_00R_01_9"
craverage n2_d1_b2061_00R_01_xbds.fits[10] output="" crmask="crmask/crmaskn2_d1_b2061_00R_01_10"
craverage n2_d1_b2061_00R_01_xbds.fits[11] output="" crmask="crmask/crmaskn2_d1_b2061_00R_01_11"
craverage n2_d1_b2061_00R_01_xbds.fits[12] output="" crmask="crmask/crmaskn2_d1_b2061_00R_01_12"
craverage n2_d1_b2061_00R_01_xbds.fits[13] output="" crmask="crmask/crmaskn2_d1_b2061_00R_01_13"
craverage n2_d1_b2061_00R_01_xbds.fits[14] output="" crmask="crmask/crmaskn2_d1_b2061_00R_01_14"
craverage n2_d1_b2061_00R_01_xbds.fits[15] output="" crmask="crmask/crmaskn2_d1_b2061_00R_01_15"
craverage n2_d1_b2061_00R_01_xbds.fits[16] output="" crmask="crmask/crmaskn2_d1_b2061_00R_01_16"
.....
```

Create an empty folder “crmask”.

**cl < crscript.cl** to run the script.

The resultant output will be masks saved within the folder “crmask”. Leave it for now.

## 6: Improving the WCS (World Coordinate System)

### A: Create Backup Files

Backup files by adding a w to the postscript. (\_xbds.w).

### B: Setting the World Coordinate System

**Run steps B, C, D, E & F for one image at a time!**

Adjust WCS information using the WCS db files provided by the KPNO webpage.

Note: at the time of writing this cookbook there was only one file available (**kp09mVtnx34.db**) for the 0.9m telescope. <http://www.noao.edu/noao/mosaic/calibs.html> **No r, R or Ha files found on the Mosaic calibration page.** We used the “V” file anyway, but ultimately found out that the database was already in the image headers:

```
# WCSASTRM = kp09m.20110119T041123 (NGC 188 V k1003) by F. Valdes 2011-01-19
```

Many of the chips were missing the WCS info. After running this task the problems went away.

!!!Run the task mscsetwcs:

**Very important: Do not skip this step!!!**

Skipping this task creates problems later with merging the chips together (mscimage)

```
PACKAGE = mscrd
TASK = mscsetwcs
```

```
images = n1_d1_b2061_00r_01_xbdsw.fits Mosaic images
database= kp09mVtnx34.db WCS database -> downloaded from Mosaic Calibration page
(ra = telra) Right ascension keyword (hours)
(dec = teldec) Declination keyword (degrees)
(equinox= telequin) Epoch keyword (years)
(ra_offs= 0.) RA offset (arcsec)
(dec_off= 0.) Dec offset (arcsec)
(extlist= )
(mode = ql)
```

## C: Coordinate Matching

You'll need to generate a coordinate list for every image in every dither. These lists will be used to achieve a better WCS match. We were working with one object field and 5 dithers in each, so we made 15 lists.

```
PACKAGE = mscrd
TASK = mscgetcatalog
```

```
input = @2062d1input List of Mosaic files
output = 2062d1coord Output file of sources
(magmin = 12.) Minimum magnitude
(magmax = 16.) Maximum magnitude
(catalog= usnob1@noao) Catalog
(rmin = 21.) Minimum radius (arcmin)
(mode = ql)
```

Input list example: 2062d1input

```
n3_d1_b2062_00r_01_xbdsw.fits
n3_d1_b2062_00R_01_xbdsw.fits
n3_d1_b2062_h16_01_xbdsw.fits
```

## D: Improve WCS with msccmatch

We were getting errors with the full list of our coordinate lists and had to shorten them. (The errors went away after shortening, but after this sometimes we got other errors – see below). You will very likely run into this problem.

Note: This task must be run on individual files.

```
mscred> epar msccmatch
```

```
input =          "n2_d2_b2061_00r_01_xbdsw.fits" List of input mosaic exposure
coords =         "d2_b2061_00r_coord_short" Coordinate file (ra/dec)
accept =         yes      Accept solution?
(outcoords =     "")      List of updated coordinate files
(usebpm =        yes)     Use bad pixel masks?
(verbose =       yes)     Verbose?\n\n# Coarse Search
(nsearch =       60)      Maximum number of positions to use in search
(search =        130)     Translation search radius (arcsec)
(rsearch =       0.3)     Rotation search radius (deg)\n\n# Fine Centroid
(cbox =          11)      Centering box (pixels)
(maxshift =      7.)      Maximum centering shift to accept (arcsec)
(csig =          0.1)     Maximum centering uncertainty to accept (arcsec)
(cfrac =         0.5)     Minimum fraction of accepted centers
(listcoords =    yes)     List centered coordinates in verbose mode?\n\n#
(nfit =          50)      Min for fit (>0) or max not found (<=0)
(rms =           1.)      Maximum fit RMS to accept (arcsec)
(fitgeometry =   "general") Fitting geometry
(reject =        3.)      Fitting rejection limit (sigma)
(update =        yes)     Update coordinate systems?
(interactive =   yes)     Interactive?
(fit =           yes)     Interactive fitting?
(graphics =      "stdgraph") Graphics device
(cursor =        "")      Graphics cursor\n
(mode =          "ql")
accept =         yes      Accept solution?
(mode =          ql)
```

```
:go
```

Will bring up a window in xterm with y vs. x. Press “x” to get x-residuals vs. x. Press “y” to repeat the process in y. Press “d” to delete the outliers (outside +/- 1). Press “f” to refit. “q” to quit. Yes to confirm match.

x	d	f
y	d	f
q		

```
MSCCMATCH:
  n4_d5_b2061_00r_01_xbdsw.fits:
    213 input coordinates
ERROR: floating point overflow
```

Copy the coordinate list. Shorten the coordinate list “d2\_b2061\_00r\_coord” and rename it “d2\_b2061\_00r\_coord\_short”. Continue to shorten until the error goes away.

```
MSCCMATCH:
  n4_d5_b2061_00r_01_xbdsw.fits:
Warning: Too few input coordinates for minimum number to fit
ERROR: MSCCMATCH failed for n4_d5_b2061_00r_01_xbdsw.fits
```

Change **nfit** from 50 to a value lower than the amount of coordinates in the shortened list.

```
MSCCMATCH:
  n4_d5_b2061_00r_01_xbdsw.fits:
    213 input coordinates
ERROR: floating point divided by zero
```

Increase **search** from 130 to 200. You might also have to decrease **nsearch** from 60 to a lower value until the error goes away.  
You might also need to change **nfit** from 50 to a value lower than **nsearch**.

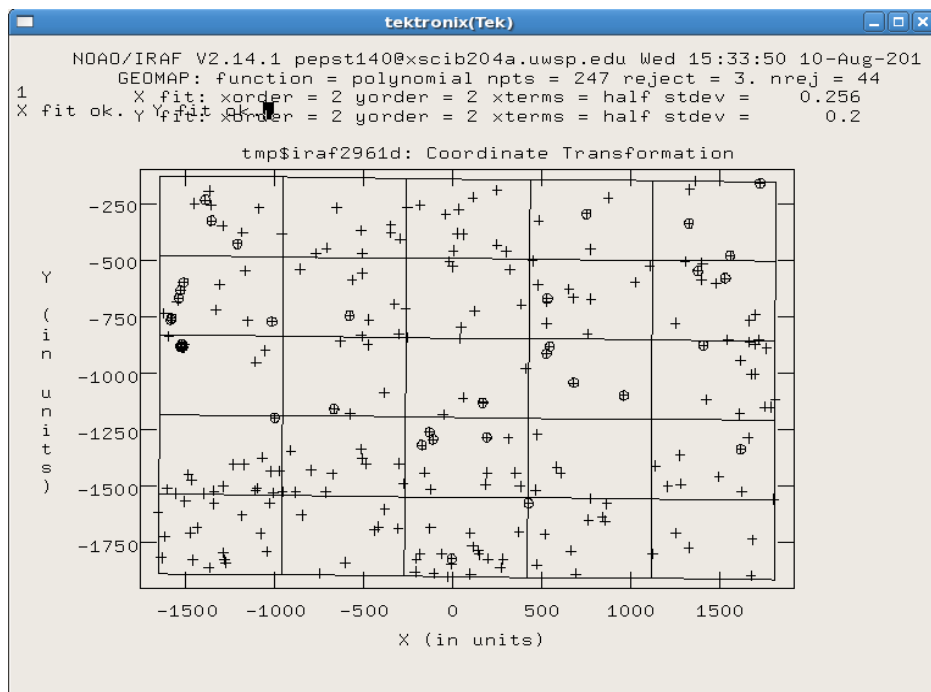
```
MSCCMATCH:
  n4_d3_b2061_00r_01_xbdsw.fits:
    141 input coordinates
    40/141 coordinates out of bounds
    search using up to 20 objects:
    search found offsets of (-228, 279) pixels and rotation -0.35 degrees
    141 input coordinates
    38/141 coordinates out of bounds
    search using up to 20 objects:
Warning: Automatic search failed
ERROR: MSCCMATCH failed for n4_d3_b2061_00r_01_xbdsw.fits
```

Increase **search** from 130 to 200+. You might also have to increase **rsearch** from 0.3 to a higher value until the error goes away (e.g., 0.4 – see example above).

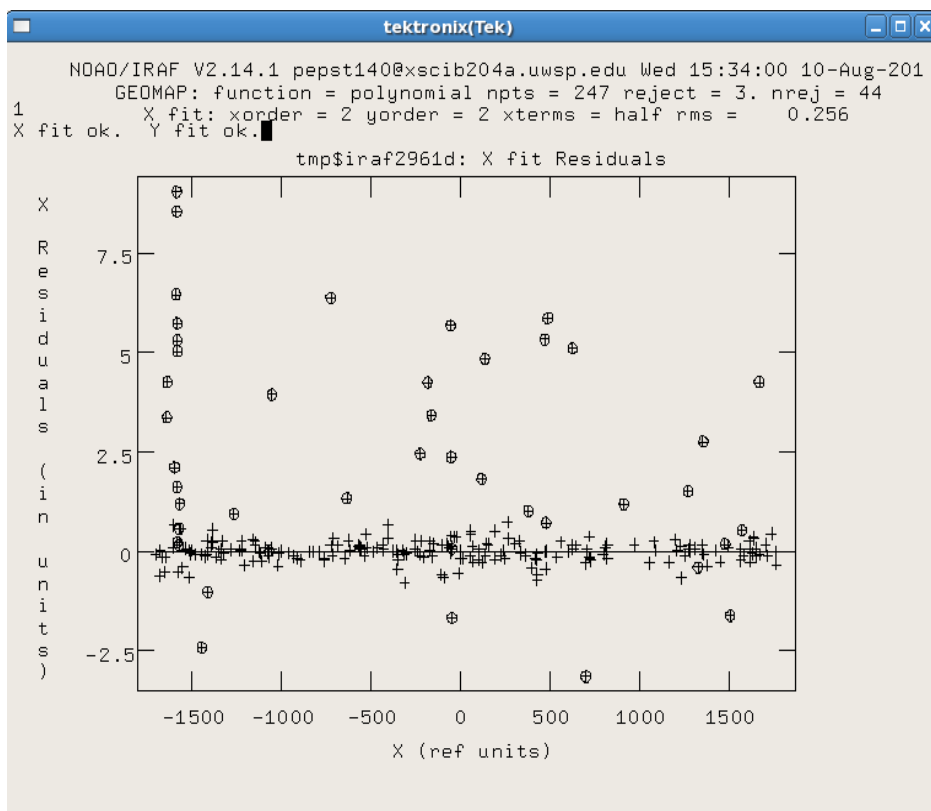
## NOTE:

If none of these works that means your coordinates are very far off. To fix this problem: copy the “.fits” file again, run **mscsetwcs** again and use a coordinate file for another filter in the same dither that worked (last one).

If this still doesn't work add an offset when you run **mscsetwcs**, then run **mscgetcatalog**, and use a coordinate file generated by **mscgetcatalog** when you run **msccmatch**.



Before “x”



After “x”



## E: mscgetcatalog Again

After you accept the solution, run `mscgetcatalog` again.

```
PACKAGE = mscred
TASK = mscgetcatalog
```

```
input = List of Mosaic files
output = d2_b2061_00r_coords_2 Output file of sources
(magmin = 12.) Minimum magnitude
(magmax = 16.) Maximum magnitude
(catalog= usnob1@noao) Catalog
(rmin = 21.) Minimum radius (arcmin)
(mode = ql)
```

Go to step F if the fit looks like the fit above (see previous page). Otherwise repeat step D. Repeat steps D & E until you can run **msccmatch** with the parameters unchanged and a full coordinate list.

## F: Checking Your Match

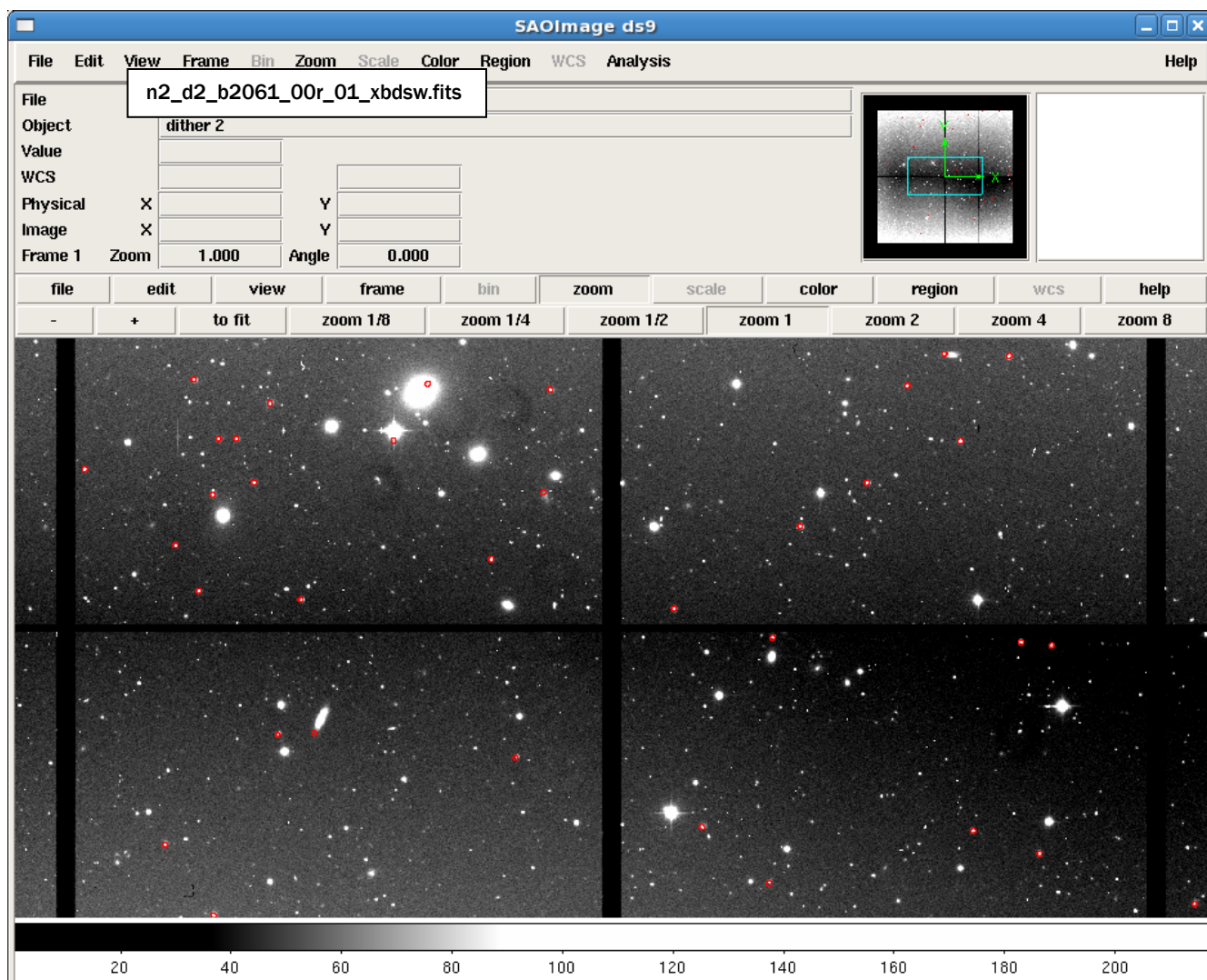
Start by displaying the image in ds9.

```
mscred> mscdisp n2_d2_b2061_00r_01_xbdsf.fits 1
```

The task `msctvmark` will lay red circles (generated by the coordinate list) over your image. If everything goes to plan you should see that they very clearly match their guide stars. **All** amplifiers should have circles centered on stars. If not, the next task (merging chips together - `mscimage`) will not work.

```
PACKAGE = mscred
TASK = msctvmark
```

```
coords = d2_b2061_00r_coords_2 List of coordinates
frame = 1 Display frame
(output = ) Output file of pixel coordinates and labels
(fields = 1,2,3) Fields for RA, DEC, and ID
(wcs = world) Coordinate type (logical|physical|world)
(mark = circle) Mark type
(radii = 10) Radii of concentric circles
(lengths= 0) Lengths and width of concentric rectangles
(font = raster) Default font
(color = 204) Gray level of marks to be drawn
(label = no) Label the marked coordinates
(nxoffse= 0) X offset in display pixels of number
(nyoffse= 0) Y offset in display pixels of number
(pointsi= 3) Size of mark type point in display pixels
(txsize = 1) Size of text and numbers in font units
(mode = ql)
```



tvmark

Note: step msczero never worked during our reduction process.

## 7: Applying Bad Pixel Masks and Cosmic Ray Masks with fixpix

### A: Create backup files

Just like before, add another letter to the postscript. (\_xbdswwf) This task will write over your files so you should back them up before any processing occurs.

```
f_copy.cl
cp n2_d1_b2061_00r_01_xbdsww.fits n2_d1_b2061_00r_01_xbdswwf.fits
cp n2_d1_b2061_00R_01_xbdsww.fits n2_d1_b2061_00R_01_xbdswwf.fits
cp n2_d1_b2061_h16_01_xbdsww.fits n2_d1_b2061_h16_01_xbdswwf.fits
cp n2_d2_b2061_00r_01_xbdsww.fits n2_d2_b2061_00r_01_xbdswwf.fits
cp n2_d2_b2061_00R_01_xbdsww.fits n2_d2_b2061_00R_01_xbdswwf.fits
cp n2_d2_b2061_h16_01_xbdsww.fits n2_d2_b2061_h16_01_xbdswwf.fits
cp n2_d3_b2061_00r_01_xbdsww.fits n2_d3_b2061_00r_01_xbdswwf.fits
.....
```

## B: Applying the BPM

Next, take your newly created image files and move each to their corresponding bpm mask folder.

```
mscred> ls bpm*
```

**bpm\_n2\_d1\_b2061\_00r\_01\_xbds:**

```
bpm_im10.pl bpm_im15.pl bpm_im4.pl bpm_im9.pl
bpm_im11.pl bpm_im16.pl bpm_im5.pl d1_00rlist
bpm_im12.pl bpm_im1.pl bpm_im6.pl d1_00rlist~
bpm_im13.pl bpm_im2.pl bpm_im7.pl n2_d1_b2061_00r_01_xbdswwf.fits
bpm_im14.pl bpm_im3.pl bpm_im8.pl
```

**bpm\_n2\_d1\_b2061\_00R\_01\_xbds:**

```
bpm_im10.pl bpm_im15.pl bpm_im4.pl bpm_im9.pl
bpm_im11.pl bpm_im16.pl bpm_im5.pl n2_d1_b2061_00R_01_xbdswwf.fits
bpm_im12.pl bpm_im1.pl bpm_im6.pl
bpm_im13.pl bpm_im2.pl bpm_im7.pl
bpm_im14.pl bpm_im3.pl bpm_im8.pl
```

The actual application of the mask will be performed within those folders.

.....

You'll have to make a chips list for each folder as fixpix can't work on a whole mosaic image at once.

EX: (d100rlist.list)

```
n2_d1_b2061_00r_01_xbdswwf.fits[im1]
n2_d1_b2061_00r_01_xbdswwf.fits[im2]
n2_d1_b2061_00r_01_xbdswwf.fits[im3]
n2_d1_b2061_00r_01_xbdswwf.fits[im4]
n2_d1_b2061_00r_01_xbdswwf.fits[im5]
n2_d1_b2061_00r_01_xbdswwf.fits[im6]
n2_d1_b2061_00r_01_xbdswwf.fits[im7]
```

.....

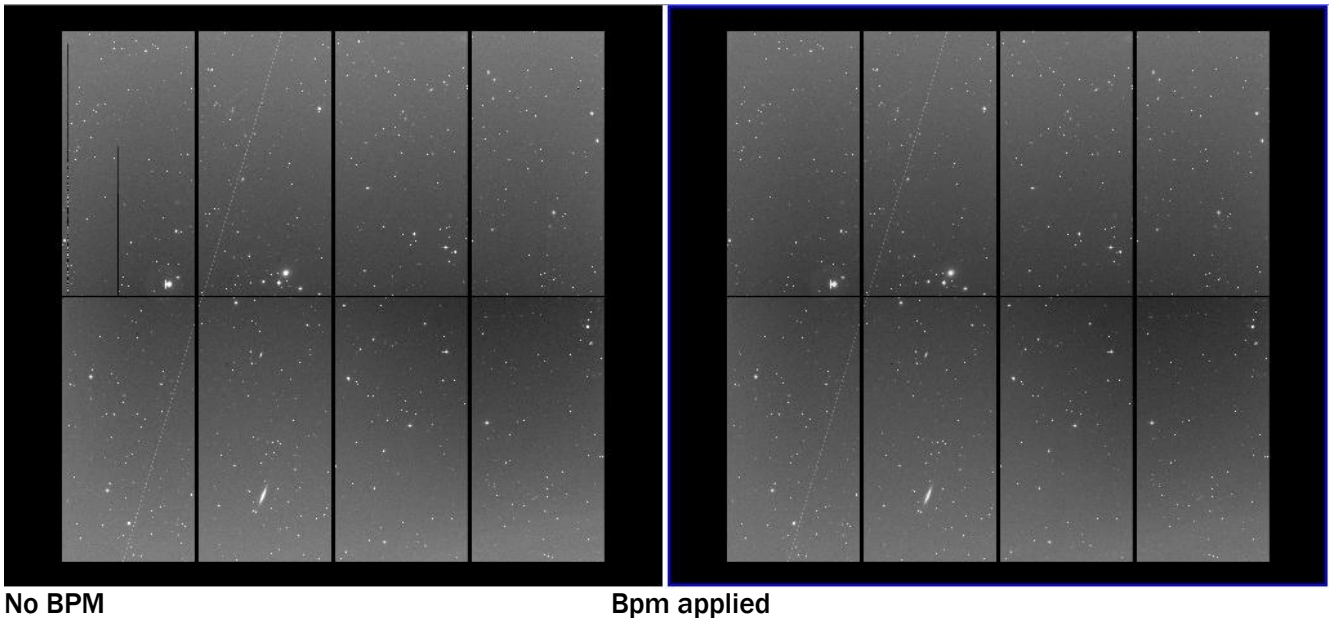
```
cd bpm_n2_d1_b2061_00r_01_xbds
```

fixpix is part of the *proto* package. So, before you run the task just type proto into the IRAF prompt and it will take you to that package.

```
mscred> epar fixpix
  images = "@d100rlist"   List of images to be fixed (list refers to one image, 16 chips)
  masks = "bpm"          List of bad pixel masks
(linterp = "INDEF")      Mask values for line interpolation
(cinterp = "INDEF")      Mask values for column interpolation
(verbose = no)           Verbose output?
(pixels = no)           List pixels?
(mode = "ql")
```

:go      Run this task in every bpm folder

When we display the image we got an error  
“Error: cannot open “bpm\_im10.pl” That’s OK.



### C: Create backup files again

Start by creating another round of backup files with postscript \_xbdswwff. Then, move the new files into the crmask folder created earlier. Finally, make another chips list. This time, include all the images as you'll be performing this round of fixpix on all of them at once.

EX

@2062list\_xbdswwff list:

```
n3_d1_b2062_00r_01_xbdswwff.fits[im1]
n3_d1_b2062_00r_01_xbdswwff.fits[im2]
n3_d1_b2062_00r_01_xbdswwff.fits[im3]
n3_d1_b2062_00r_01_xbdswwff.fits[im4]
n3_d1_b2062_00r_01_xbdswwff.fits[im5]
n3_d1_b2062_00r_01_xbdswwff.fits[im6]
n3_d1_b2062_00r_01_xbdswwff.fits[im7]
n3_d1_b2062_00r_01_xbdswwff.fits[im8]
n3_d1_b2062_00r_01_xbdswwff.fits[im9]
```

.....

### D: Applying the Cosmic Ray mask (crmask)

Create another chips list for the masks. Make sure the two lists match.

EX

@crlist list:

```
crmaskn3_d1_b2062_00r_01_1.fits[im1]
crmaskn3_d1_b2062_00r_01_2.fits[im2]
crmaskn3_d1_b2062_00r_01_3.fits[im3]
crmaskn3_d1_b2062_00r_01_4.fits[im4]
crmaskn3_d1_b2062_00r_01_5.fits[im5]
crmaskn3_d1_b2062_00r_01_6.fits[im6]
```

```

crmaskn3_d1_b2062_00r_01_7.fits[im7]
crmaskn3_d1_b2062_00r_01_8.fits[im8]
crmaskn3_d1_b2062_00r_01_9.fits[im9]
crmaskn3_d1_b2062_00r_01_10.fits[im10]
crmaskn3_d1_b2062_00r_01_11.fits[im11]
crmaskn3_d1_b2062_00r_01_12.fits[im12]
crmaskn3_d1_b2062_00r_01_13.fits[im13]
.....

```

Run fixpix again with the following parameters.

```

PACKAGE = proto
TASK = fixpix

```

```

images = @2062list_xbdswwff List of images to be fixed
masks = @crlst List of bad pixel masks
(linterp= INDEF) Mask values for line interpolation
(cinterp= INDEF) Mask values for column interpolation
(verbose= yes) Verbose output?
(pixels = no) List pixels?
(mode = ql)

```

## 8: Merge Chips

Make another list of your image files, this time adding an “i” to your postscript for image. Don't make copies of the files as the process will do it for you with an output list option (unlike the fixpix task). Mscimage is pretty straight forward task, but can take a long time. It's basically turning your 8 amplifier images into a single fits image file and creating the masks to cover the spaces between the chips.

```

PACKAGE = mscrd
TASK = mscimage

```

```

input = @2062list_xbdswwff List of input mosaic exposures
output = @2062list_xbdswwffi List of output images
(format = image) Output format (image|mef)
(pixmask= yes) Create pixel mask?
(verbose= yes) Verbose output?

# Output WCS parameters
(wcssour= image) Output WCS source (image|parameters|match)
(referen= ) Reference image
(ra = INDEF) RA of tangent point (hours)
(dec = INDEF) DEC of tangent point (degrees)
(scale = INDEF) Scale (arcsec/pixel)
(rotatio= INDEF) Rotation of DEC from N to E (degrees)

# Resampling parameters
(blank = 0.) Blank value
(interpo= sinc17) Interpolant for data
(minterp= linear) Interpolant for mask
(boundar= constant) Boundary extension

```

```
(constan=      0.) Constant boundary extension value
(fluxcon=      no) Preserve flux per unit area?
(ntrim  =      7) Edge trim in each extension
(nxblock=     4200) X dimension of working block size in pixels
(nyblock=     2100) Y dimension of working block size in pixels
```

```
# Geometric mapping parameters
(interac=      no) Fit mapping interactively?
(nx   =      10) Number of x grid points
(ny   =      20) Number of y grid points
(fitgeom=     general) Fitting geometry
(xxorder=      4) Order of x fit in x
(xyorder=      4) Order of x fit in y
(xxterms=     half) X fit cross terms type
(yxorder=      4) Order of y fit in x
(yyorder=      4) Order of y fit in y
(yxterms=     half) Y fit cross terms type
```

```
(fd_in  =      )
(fd_ext =      )
(fd_coor=      )
(mode   =      ql)
```

@2061list\_xbdswwffi list:

```
n2_d1_b2061_00r_01_xbdswwffi.fits
n2_d1_b2061_00R_01_xbdswwffi.fits
n2_d1_b2061_h16_01_xbdswwffi.fits
n2_d2_b2061_00r_01_xbdswwffi.fits
```

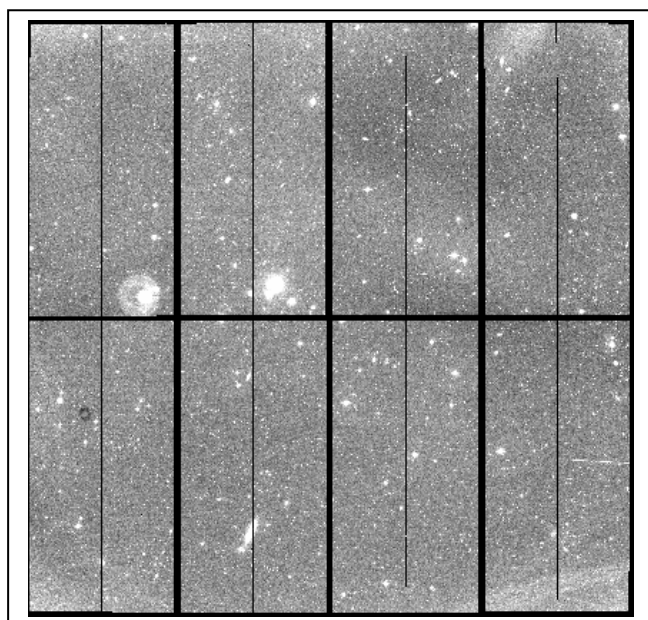
.....

masks automatically named:

```
n2_d4_b2061_h16_01_xbdswwffi_bpm.pl
```

.....

(Masks can be viewed in DS9 with *mscdisp n2\_d4\_b2061\_h16\_01\_xbdswwffi\_bpm.pl 1*)



\_xbdswwffi image

## 9: Removing the Sky Gradient with mscskysub

As always, begin by making input/output lists. Then, simply run the task. At the beginning we had some trouble with this part of the process (using the Legendre order2 function for fit, as recommended by the Januzzi cookbook) and were left with some large scale gradients in our remaining frames. Using the spline3 order4 function gave us the best result for removing the large scale gradient.

```
PACKAGE = mscrcd
TASK = mscskysub
```

```
input  = @2062list_xbdswwffi  Input images to be fit
output = @2062list_xbdswwffis  Output images
xorder = 4  Order of function in x
yorder = 4  Order of function in y
(type_ou= residual) Type of output (fit,residual,response,clean)
(function= spline3) Function to be fit (legendre,chebyshev,spline3)
(cross_t= yes) Include cross-terms for polynomials?
(xmedian= 100) X length of median box
(ymedian= 100) Y length of median box
(median_ = 50.) Minimum fraction of pixels in median box
(lower = 0.) Lower limit for residuals
(upper = 0.) Upper limit for residuals
(ngrow = 0) Radius of region growing circle
(niter = 0) Maximum number of rejection cycles → used 3 for data from Spring 2012
(regions= mask) Good regions (all,rows,columns,border,sections,c
(rows = *) Rows to be fit
(columns= *) Columns to be fit
(border = 100) Width of border to be fit
(section= ) File name for sections list
(circle = ) Circle specifications
(mask = BPM) Mask
(div_min= INDEF) Division minimum for response output
(mode = ql)
```

Notes: For data taken in Spring 2012 Legendre order 4 gave a better fit.

Not significant improvement after the mscskyfit for field 206-1.

Field 206-3 had two big nearby galaxies overlapping the field, making the fit impossible, so this step was skipped for that field.

## 10: Improving the WCS further with msccmatch

(You can skip this step if your WCS is perfect already)

### A: Run mscgetcatalog again

Start by generating a coordinate file for **every dither**. Choose a filter and run mscgetcat on every dither in that filter. We chose h16 and ended up with 5 coordinate files.

```
PACKAGE = mscrcd
TASK = mscgetcatalog
```

```
input  = n3_d5_b2062_h16_01_xbdswwffis.fits  List of Mosaic files
output = 2062d5  Output file of sources
(magmin = 12.) Minimum magnitude
(magmax = 16.) Maximum magnitude
```



```
(catalog=      usnob1@noao) Catalog
(rmin  =      21.) Minimum radius (arcmin)
(mode  =      ql)
```

Note: usnob1@noao stopped working for us. We switched to the NOAO:USNO-A2 catalog and everything went fine from there.

Only if usnob1@noao is not working use NOAO:USNO-A2 !!!

```
mscred> mscgetcat
List of Mosaic files (n3_d3_b2062_00r_01_xbdswwffis.fits): n3_d1_b2062_h16_01_xbdswwffis.fits
Output file of sources (2062d3): 2062d1
mscred> mscgetcat
List of Mosaic files ( n3_d1_b2062_h16_01_xbdswwffis.fits): n3_d2_b2062_h16_01_xbdswwffis.fits
Output file of sources (2062d1): 2062d2
mscred> mscgetcat
List of Mosaic files (n3_d2_b2062_h16_01_xbdswwffis.fits): n3_d3_b2062_h16_01_xbdswwffis.fits
Output file of sources (2062d2): 2062d3
mscred> mscgetcat
List of Mosaic files (n3_d3_b2062_h16_01_xbdswwffis.fits): n3_d4_b2062_h16_01_xbdswwffis.fits
Output file of sources (2062d3): 2062d4
mscred> mscgetcat
List of Mosaic files (n3_d4_b2062_h16_01_xbdswwffis.fits): n3_d5_b2062_h16_01_xbdswwffis.fits
Output file of sources (2062d4): 2062d5
```

## B: Run msccmatch again

After you have the files, run msccmatch on every single image including the files you used to generate the coordinate lists.

```
PACKAGE = mscred
TASK = msccmatch
```

```
input  = n3_d5_b2062_h16_01_xbdswwffis.fits List of input mosaic exposures
coords = 2062d5 Coordinate file (ra/dec)
(outcoo= ) List of updated coordinate files
(usebpm = yes) Use bad pixel masks?
(verbose= yes) Verbose?
```

### # Coarse Search

```
(nsearch= 60) Maximum number of positions to use in search
(search  = 130.) Translation search radius (arcsec)
(rsearch= 0.3) Rotation search radius (deg)
```

### # Fine Centroiding

```
(cbox  = 11) Centering box (pixels)
(maxshif= 7.) Maximum centering shift to accept (arcsec)
(csig  = 0.1) Maximum centering uncertainty to accept (arcsec)
(cfrac = 0.5) Minimum fraction of accepted centers
(listcoo= yes) List centered coordinates in verbose mode?
```

### # WCS Fitting

```
(nfit  = 50) Min for fit (>0) or max not found (<=0)
(rms   = 1.) Maximum fit RMS to accept (arcsec)
(fitgeom= general) Fitting geometry
(reject = 3.) Fitting rejection limit (sigma)
(update = yes) Update coordinate systems?
(interac= yes) Interactive?
```



```
(fit = yes) Interactive fitting?
(graphic= stdgraph) Graphics device
(cursor = ) Graphics cursor

accept = yes Accept solution?
(mode = ql)
```

x	d	f
y	d	f
q		

## 11: Create backup files

Copy \_xbdswwffis.fits → \_xbdswwffisw.fits (create a copy script .cl)

## 12: Determine the Relative Scalings of Different Images

### A: Run getcat Again

You're going to run getcat on one image per filter. The goal is to match the coordinates from all dithers to one image. Choose one image (the most central one-d5) in each filter as a reference and run getcat on that image. We ended up with three coordinate files: 00rref, 00Rref, and h16ref.

```
PACKAGE = mscred
TASK = mscgetcatalog
```

```
input = n3_d5_b2062_00r_01_xbdswwffis.fits List of Mosaic files
output = 00rref Output file of sources
(magmin = 12) Minimum magnitude
(magmax = 16) Maximum magnitude
(catalog= usnob1@noao) Catalog
(rmin = 21.) Minimum radius (arcmin)
(mode = ql)
```

### B: Run mscimatch

mscimatch uses that coordinate list that you just created to calibrate the scalings required by every dither for stacking. Make sure to include your most recent set of bpm's in your working folder.

#### h16list example

```
n3_d1_b2062_h16_01_xbdswwffisw.fits
n3_d2_b2062_h16_01_xbdswwffisw.fits
n3_d3_b2062_h16_01_xbdswwffisw.fits
n3_d4_b2062_h16_01_xbdswwffisw.fits
n3_d5_b2062_h16_01_xbdswwffisw.fits
```

```
PACKAGE = mscred
TASK = mscimatch
```

```
input = @h16list List of images
coords = h16ref File of coordinates
(bpm = BPM) List of bad pixel masks
(measure= ) Measurment file
(scale = yes) Determine scale?
```

(zero = **no**) Determine zero offset? → **yes** only if you skip mscskysub  
 (box1 = 21) Inner box size for statistics  
 (box2 = 51) Outer box size for statistics  
 (lower = -100.) Lower limit for good data  
 (upper = INDEF) Upper limit for good data  
 (niterat= 3) Number of sigma clipping iterations  
 (sigma = 3.) Sigma clipping factor  
 (interac= **yes**) Interactive?  
 (verbose= **yes**) Verbose?  
 accept = **yes** Accept scaling and update images?  
 (mode = ql)

? Print options

SET GRAPH WINDOW

w a Autoscale x and y axes

w z Zoom x and y axes about cursor

d Delete data point nearest the cursor

f Fit the data and redraw or overplot

q Go to next image & at the end exit the interactive curve fitting.

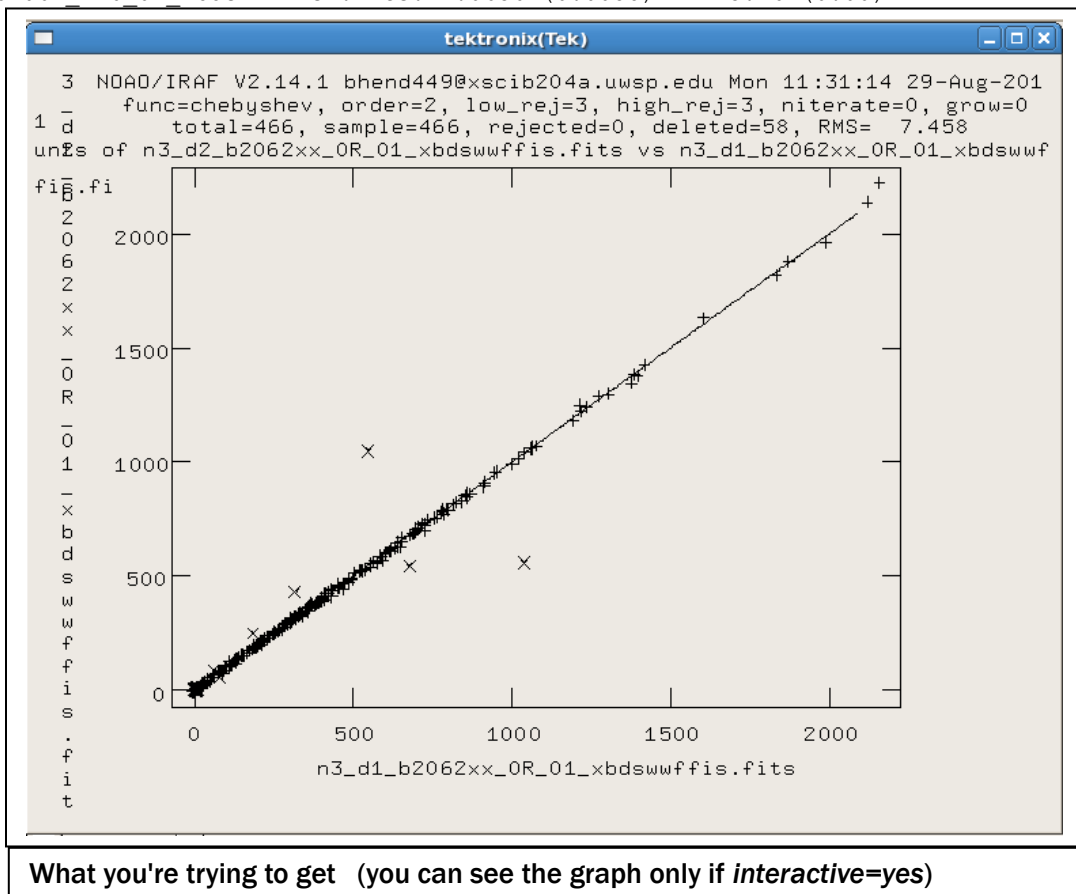
#### MSCIMATCH:

```

Reading region coordinates from h16ref
212 coordinates read
Measuring regions in n4_d1_b2061_h16_01_xbdswwffisw.fits ...
Using bad pixel mask n4_d1_b2061_h16_01_xbdswwffi_bpm ...
103 good regions measured
Measuring regions in n4_d2_b2061_h16_01_xbdswwffisw.fits ...
Using bad pixel mask n4_d2_b2061_h16_01_xbdswwffi_bpm ...
110 good regions measured
Measuring regions in n4_d3_b2061_h16_01_xbdswwffisw.fits ...
Using bad pixel mask n4_d3_b2061_h16_01_xbdswwffi_bpm ...
110 good regions measured
Measuring regions in n4_d4_b2061_h16_01_xbdswwffisw.fits ...
Using bad pixel mask n4_d4_b2061_h16_01_xbdswwffi_bpm ...
103 good regions measured
Measuring regions in n4_d5_b2061_h16_01_xbdswwffisw.fits ...
Using bad pixel mask n4_d5_b2061_h16_01_xbdswwffi_bpm ...
97 good regions measured
Determining scale factors ...
n4_d1_b2061_h16_01_xbdswwffisw.fits: 1.0000 (0.0000) 0.00 (0.00)
n4_d2_b2061_h16_01_xbdswwffisw.fits: 1.0158 (0.0088) -6.62 (0.00)
n4_d3_b2061_h16_01_xbdswwffisw.fits: 1.0473 (0.0133) -12.87 (0.00)
n4_d4_b2061_h16_01_xbdswwffisw.fits: 1.0525 (0.0051) -17.38 (0.00)
n4_d5_b2061_h16_01_xbdswwffisw.fits: 1.0636 (0.0058) -23.76 (0.00)

```

If the night was photometric all the values should be around 1.



## 13: Generate the Stacked Image with mscstack

The task itself should feel pretty standard by now, just run it as follows. Output will be the name of your stacked image.

PACKAGE = mscrcd

TASK = mscstack

```
input =      @h16list List of images to combine
output =     h16med) Output image      → h16av in case you use average
(headers=    ) List of header files (optional)
(bpmasks=   bpm_h16) List of bad pixel masks (optional)
(rejmask=    ) List of rejection masks (optional)
(nrejmas=    ) List of number rejected masks (optional)
(expmask=    ) List of exposure masks (optional)
(sigmals =   ) List of sigma images (optional)

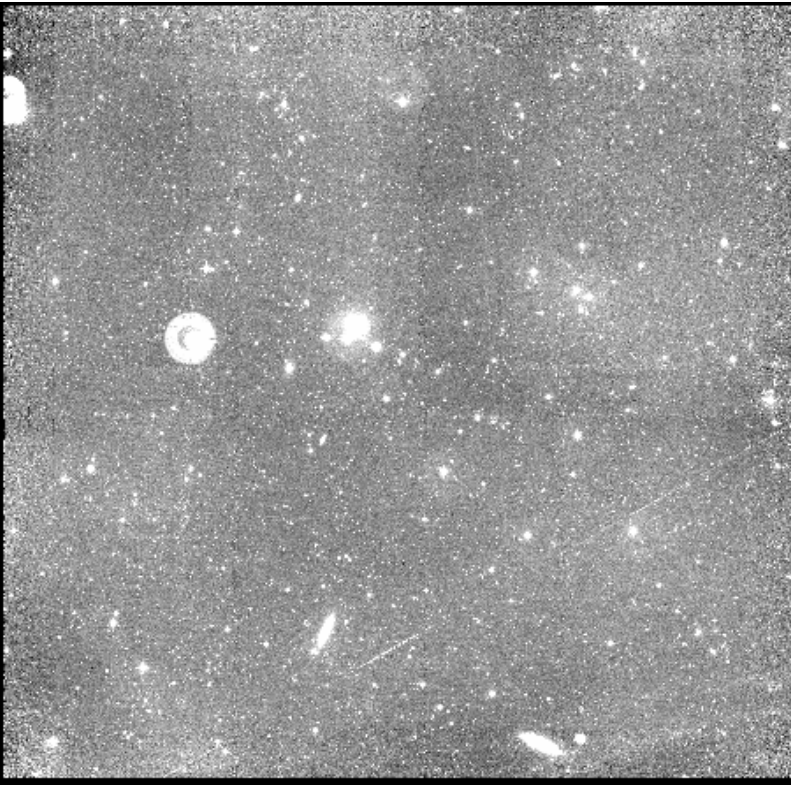
(combine=     median) Type of combine operation (median|average) → average worked better for Spring '12 data
(reject =     avsigclip) Type of rejection
(masktyp=     goodvalue) Mask type
(maskval=     0.) Mask value
(blank =      0.) Value if there are no pixels

(scale =      !mscscale) Image scaling
(zero =       !msczero) Image zero point offset
(weight =     none) Image weights
(statsec=     ) Image section for computing statistics

(lthresh=     1.) Lower threshold
(hthresh=     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 =     5.) Lower sigma clipping factor
(hsigma =     2.9) Upper sigma clipping factor
(rdnoise=     rdnoise) ccdclip: CCD readout noise (electrons)
(gain =       gain) ccdclip: CCD gain (electrons/DN)
(snoise =     0.) ccdclip: Sensitivity noise (fraction)
(sigscal=     0.1) Tolerance for sigma clipping scaling corrections
(pclip =     -0.5) pclip: Percentile clipping parameter
(grow =       0.) Radius (pixels) for neighbor rejection
(mode =       ql)
```

Note: Use ccdclip instead of avsigclip if your RDNOISE and GAIN have good values in the header.

Now, if you're really lucky your final image looks amazing. If not, don't panic, this is exactly why you've been backing up your images throughout the entire process. Think on the problem for a while. You've likely had to solve many issues while running through this ordeal. Our final images looked terrible at first, but we were able to go back a few steps and correct a few things. The biggest change was an additional round of mscmatch after mscimage. The WCS coordinates in our initial headers, it seems, weren't the greatest. Running mscmatch that additional time really helped our final stack, which turned out much much better.



Stacked image h16av.fits opened with msdisp  
(if you open the image directly from ds9 it will look very smooth)

## 14: Apply the bad pixel mask to the stacked image

```
PACKAGE = proto
TASK = fixpix

images = h16av List of images to be fixed
masks = bpm_h16av List of bad pixel masks
(linterp= INDEF) Mask values for line interpolation
(cinterp= INDEF) Mask values for column interpolation
(verbose= yes) Verbose output?
(pixels = no) List pixels?
(mode = ql)
```

After applying the bpm mask the black lines and black dots of bad pixels would disappear.

## 15: Bring sky to zero

Use imarith to subtract the background sky to zero. The SKYMEAN count value is found in the header, or you can generate the MODE value (which is the most common value in the image a.k.a., the sky) by running imstat. Double check the sky value by scrolling the cursor over the image in ds9. SKYMEAN is more reliable than the MODE.

Note: To get the SKYMEAN parameter open the image in ds9 with “open”, then “display header”.

If you skip mscskysub the SKYMEAN parameter is not in the header. Use a clean rectangular area on the image to find the sky value with imstat.

```
mscired> imstat 00rav.fits[280:730,4983:5158]
#          IMAGE          NPIX      MEAN      STDDEV      MIN      MAX      MODE
00rav.fits[280:730,4983:5158]      79376      423.7      13.02      353.8      970.      422.4
```

```
mscred> imarith 00rav.fits - 366.8206 00rav_s.fits
mscred> imarith 00Rav.fits - 598.7870 00Rav_s.fits
mscred> imarith h16av.fits - 100.7042 h16av_s.fits
```

After subtracting the sky double check the sky value by scrolling the cursor over the image in ds9. The sky should be around zero (both positive and negative values).

**Note:** If you use the SKYMEAN usually the MSCZERO is zero in the header:

```
SKYMEAN = 366.8206
RADECSYS= 'FK5'
EQUINOX = 2000.
MSCSCALE= 1. →sky value is 366.8206
MSCZERO = 0.
```

If not, add the SKYMEAN and MSCZERO to find the sky value:

```
NCOMBINE= 80
SKYMEAN = 432.2816
RADECSYS= 'FK5'
EQUINOX = 2000.
MSCSCALE= 0.929925232295051 →sky value is 432.2816+166.5056=598.787
MSCZERO = 166.505571981292
```

## 16: Scale images

### A: Record Flux with imexam

To match the contribution of the continuum, you need to scale the R to the H-alpha. Use imexam to measure the flux from a few stars and compare. You'll notice that the flux is different for the same star in R and H-alpha filters.

Open ds9

```
mscred> imexam (click "a")
display frame (1:) (1):
# COL LINE COORDINATES
# R MAG FLUX SKY PEAK E PA BETA ENCLOSED MOFFAT DIRECT
7176.64 6312.73 7176.64 6312.73
9.68 11.93 168853. 1.351 9979. 0.10 -70 2.25 3.29 3.34 3.23 -> R
7178.99 6312.28 7178.99 6312.28
10.58 12.87 71256. -0.1946 3909. 0.11 -64 9.03 3.61 3.68 3.53 -> ha
7755.85 6301.16 7755.85 6301.16
7.13 13.38 44526. -5. 3143. 0.12 -58 0.66 2.77 2.58 2.38 ->R
7758.23 6300.65 7758.23 6300.65
14.25 19917. -2.172 1067. 0.14 -53 6.58 3.58 3.71 3.65 -> ha
...
```

(The counts in R are higher than H-alpha)

```
71256/168853=0.422
19917/44526=0.447
...
```

Average=0.420 (for 5 stars)

## B: Use imarith to scale the R to H-alpha

```
mscred> imarith 00Rmed_s.fits * .42 00Rscaled.fits
```

## 17: Final Subtraction

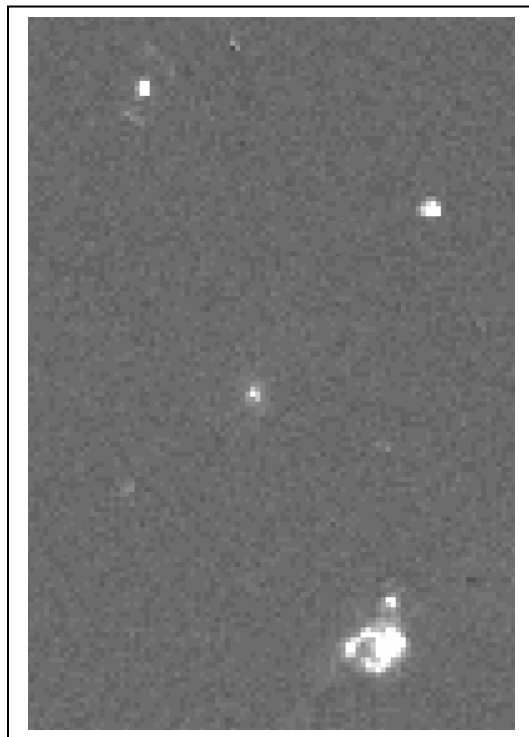
```
mscred> imarith h16med_s.fits - 00Rscaled.fits ha.fits
```

Well that was easy. Congratulations on producing your final image. If the subtracted stars are uniformly offset in any direction you can always run imshift on one of the stacked images to manually align them before subtracting.

**Note:** If you get this error “Input images have different dimensions” you need to bring the images to the same dimension:

```
mscred> imhe 00Rscaled.fits
00Rscaled.fits[8964,8788][real]: NRGb206-1_R
mscred> imhe h16av_s.fits
h16av_s.fits[8963,8787][real]: NRGb206-1_ha

mscred> imcopy 00Rscaled.fits[1:8963,1:8787] 00Rscaledcut.fits
00Rscaled.fits[1:8963,1:8787] -> 00Rscaledcut.fits
mscred> imhe 00Rscaledcut.fits
00Rscaledcut.fits[8963,8787][real]: NRGb206-1_R
```



## 18: Shift Correction (OPTIONAL)

If your final image looks blurry, or there are “hills and pits” in your counts, then the R and H-alpha images are likely incorrectly aligned.

### A: measure the shift in x & y with imexam

```
msscred> imexam (click “a”)
display frame (1:) (1):
# COL LINE COORDINATES
# R MAG FLUX SKY PEAK E PA BETA ENCLOSED MOFFAT DIRECT
3698.25 4449.37 3698.25 4449.37 --> R
10.04 10.64 553960. 3.162 32455. 0.04 -89 2.40 3.40 3.40 3.35
3699.50 4448.54 3699.50 4448.54 --> ha
11.73 11.56 237813. -1.807 11820. 0.09 -73 12.9 3.77 3.93 3.91
4198.94 4594.31 4198.94 4594.31
9.33 11.03 386290. 1.517 23002. 0.08 -56 1.96 3.16 3.18 3.11
4200.32 4593.67 4200.32 4593.67
11.99 159694. -2.442 8635. 0.09 -64 33.0 3.62 3.82 3.75
```

3698.25-3699.50=-1.25

4449.37-4448.54=0.83

Average shift in x & y for a few stars: -1.31

(Match the coordinates of ha to R)

### B: Shift one image relative to the other

PACKAGE = immatch

TASK = imshift

input = 00Rscaledcut.fits Input images to be fit

output = 00Rscaledcutsh.fits Output images

xshift = 1.31 Fractional pixel shift in x

yshift = -0.735 Fractional pixel shift in y

(shifts\_ = ) Text file containing shifts for each image

(interp\_ = linear) Interpolant (nearest,linear,poly3,poly5,spline3,

(boundar= nearest) Boundary (constant,nearest,reflect,wrap)

(constan= 0.) Constant for boundary extension

(mode = ql)

### C: Go back to step 17 and subtract again

## 19: Image Rotation and Cleaning

Rotate with the IRAF task “rotate” or “imtranspose”. The image can be cleaned with imedit.