WIRCSoft
IDL WIRC Pipeline

Jason Surace

July 1, 2008

This is the manual for a pipeline designed to reduce data from the Palomar WIRC near-infrared camera. It is written entirely in IDL, and uses the Terapix Sextractor and SWARP software for additional functionality, and it could easily be adapted to handle other near-IR cameras with the same methodology. A camera like WIRC produced data at a truly prodigious rate, and the task of reducing what could easily be several hundred 2048x2048 frames per night exceeds what can still be done reasonably by hand, hence the need for automation. This package aims to streamline as much as possible the data reduction process, so that an entire night's data can be easily reduced in as little as a day.

With the code, you will carry out all the standard procedures: making darks and flats, sky-subtracting, and also photometric and astrometric calibration and reprojection. The code uses the technique of a running median sky – that is, every frame gets a unique sky frame generated for it, created from temporally adjacent, dithered images.

0. Installation

Installation is fairly trivial. Users are expected to have a passing familiarity with IDL. In particular, you will need to look at the tasks as the various command-line options are explicitly enumerated in the headers.

a) Download the tarball containing the IDL scripts and place them in your IDL path. You could even put them in your data directory, which would make them easy to modify.

b) You need to have *sextractor* and *swarp* in your current path. The code was developed under Sextractor 2.5.0 and SWARP 2.16.0. You will get a significant speed advantage if you have compiled the multi-processor versions (and have multiple cpus!).

c) You also need the IDL astronomy libraries installed. This was developed under IDL 6.2 and the July 2004 version of the IDL astro routines.

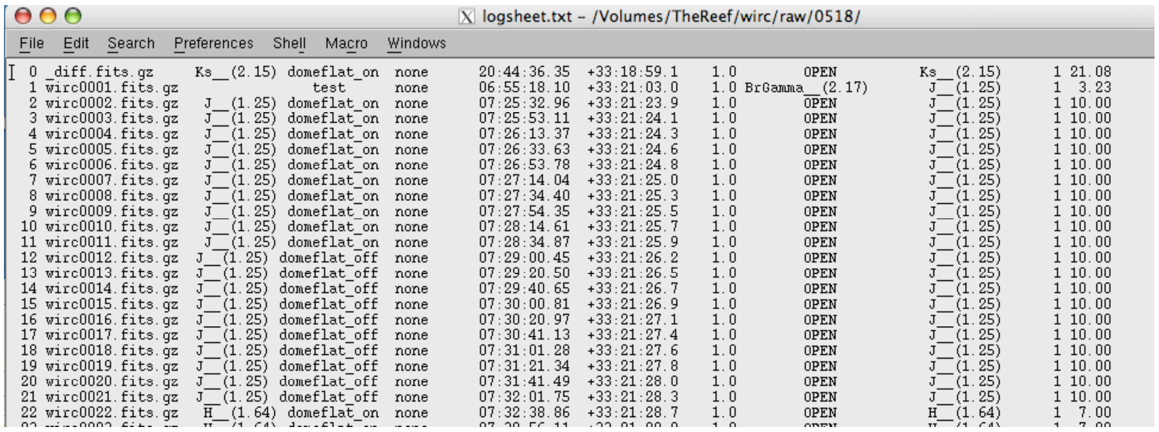
d) You will want all of your raw data to be in a single working directory. It is ok to leave them gzipped, as all of the tasks will read gzipped files, and most of them will write gzipped output if the /COMPRESS flag is used. This will save a lot of space, and adds surprisingly little overhead.

e) You will need to copy the sex_files directory to your working directory. These files are needed for the Terapix software to run (sextractor and swarp). You will want a local copy, as you may wind up editing the pointing centers in the swarp control files. Note that this is the easiest place to goof up the installation. Those control files have paths associated with them that you may need to edit depending on how sextractor was installed on your system.

1. Make a Logsheet

First, make a list of all the raw data files. Then, run a simple command to create a text file which contains useful information gleaned from the header. For purposes of calibration data flow, this file is important. The file contains for each raw image: an index number, filename, object description, object type, RA, DEC, airmass, forward filter wheel position, rear filter wheel position, number of coadds, and the exposure time.

```
humu% ls *gz > all.list
IDL> wirc_logsheet,'all.list'
humu% nedit logsheet.txt
```



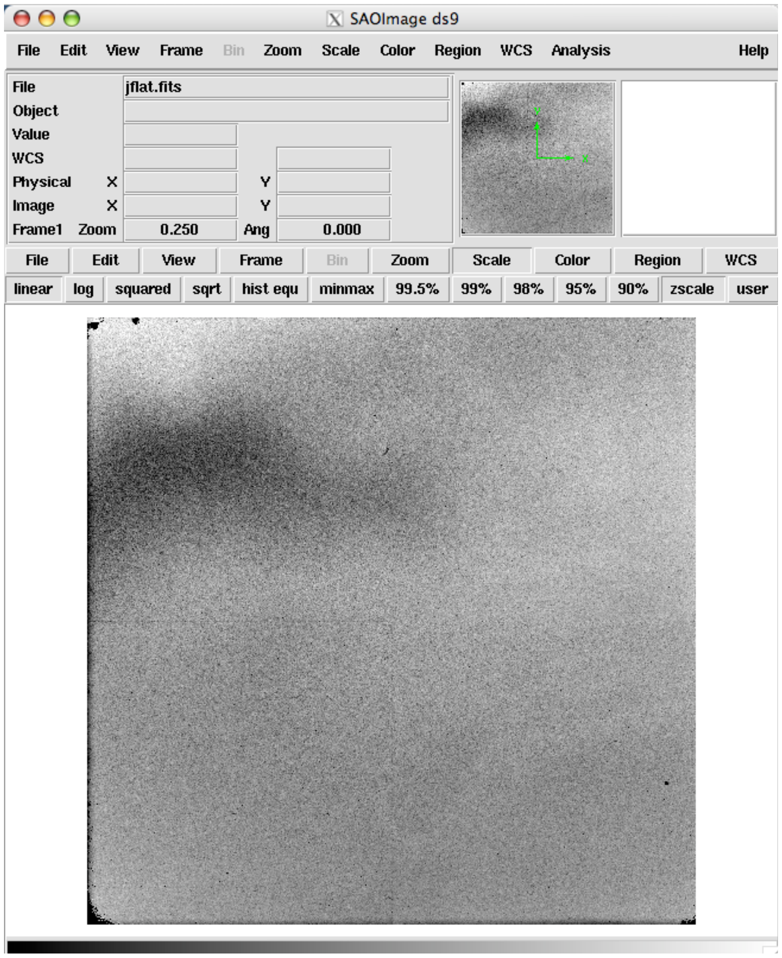
```
 0 _diff.fits.gz     Ks__(2.15) domeflat_on  none      20:44:36.35  +33:18:59.1    1.0          OPEN          Ks__(2.15)        1 21.08
 1 wirc0001.fits.gz                 test     none      06:55:18.10  +33:21:03.0    1.0 BrGamma__(2.17)         J__(1.25)        1  3.23
 2 wirc0002.fits.gz   J__(1.25) domeflat_on  none      07:25:32.96  +33:21:23.9    1.0          OPEN           J__(1.25)        1 10.00
 3 wirc0003.fits.gz   J__(1.25) domeflat_on  none      07:25:53.11  +33:21:24.1    1.0          OPEN           J__(1.25)        1 10.00
 4 wirc0004.fits.gz   J__(1.25) domeflat_on  none      07:26:13.37  +33:21:24.3    1.0          OPEN           J__(1.25)        1 10.00
 5 wirc0005.fits.gz   J__(1.25) domeflat_on  none      07:26:33.63  +33:21:24.6    1.0          OPEN           J__(1.25)        1 10.00
 6 wirc0006.fits.gz   J__(1.25) domeflat_on  none      07:26:53.78  +33:21:24.8    1.0          OPEN           J__(1.25)        1 10.00
 7 wirc0007.fits.gz   J__(1.25) domeflat_on  none      07:27:14.04  +33:21:25.0    1.0          OPEN           J__(1.25)        1 10.00
 8 wirc0008.fits.gz   J__(1.25) domeflat_on  none      07:27:34.40  +33:21:25.3    1.0          OPEN           J__(1.25)        1 10.00
 9 wirc0009.fits.gz   J__(1.25) domeflat_on  none      07:27:54.35  +33:21:25.5    1.0          OPEN           J__(1.25)        1 10.00
10 wirc0010.fits.gz   J__(1.25) domeflat_on  none      07:28:14.61  +33:21:25.7    1.0          OPEN           J__(1.25)        1 10.00
11 wirc0011.fits.gz   J__(1.25) domeflat_on  none      07:28:34.87  +33:21:25.9    1.0          OPEN           J__(1.25)        1 10.00
12 wirc0012.fits.gz  J__(1.25) domeflat_off  none      07:29:00.45  +33:21:26.2    1.0          OPEN           J__(1.25)        1 10.00
13 wirc0013.fits.gz  J__(1.25) domeflat_off  none      07:29:20.50  +33:21:26.5    1.0          OPEN           J__(1.25)        1 10.00
14 wirc0014.fits.gz  J__(1.25) domeflat_off  none      07:29:40.65  +33:21:26.7    1.0          OPEN           J__(1.25)        1 10.00
15 wirc0015.fits.gz  J__(1.25) domeflat_off  none      07:30:00.81  +33:21:26.9    1.0          OPEN           J__(1.25)        1 10.00
16 wirc0016.fits.gz  J__(1.25) domeflat_off  none      07:30:20.97  +33:21:27.1    1.0          OPEN           J__(1.25)        1 10.00
17 wirc0017.fits.gz  J__(1.25) domeflat_off  none      07:30:41.13  +33:21:27.4    1.0          OPEN           J__(1.25)        1 10.00
18 wirc0018.fits.gz  J__(1.25) domeflat_off  none      07:31:01.28  +33:21:27.6    1.0          OPEN           J__(1.25)        1 10.00
19 wirc0019.fits.gz  J__(1.25) domeflat_off  none      07:31:21.34  +33:21:27.8    1.0          OPEN           J__(1.25)        1 10.00
20 wirc0020.fits.gz  J__(1.25) domeflat_off  none      07:31:41.49  +33:21:28.0    1.0          OPEN           J__(1.25)        1 10.00
21 wirc0021.fits.gz  J__(1.25) domeflat_off  none      07:32:01.75  +33:21:28.3    1.0          OPEN           J__(1.25)        1 10.00
22 wirc0022.fits.gz   H__(1.64) domeflat_on  none      07:32:38.86  +33:21:28.7    1.0          OPEN           H__(1.64)        1  7.00
```

2. Make Flatfields

The following IDL command will make the dome flats. It is assumed they were taken as a sequence of lamp-on, then lamp-off images. The two vectors are the indices in the logsheet. The code will automatically name the flat appropriately so that the rest of the commands find the file automatically. It only recognizes J, H, and Kshort. For other flats you can specify a name on the command line (see the header of the IDL routine). For each image, the median counts are given, which is handy for making sure these really are the lamp-on and lamp-off images.

```
IDL> wirc_makeflat,'all.list',[2,11],[12,21],/VERBOSE
% READCOL: 397 valid lines read
Dome On:
wirc0002.fits.gz     12369.0
wirc0003.fits.gz     12366.0
wirc0004.fits.gz     12357.0
wirc0005.fits.gz     12337.0
wirc0006.fits.gz     12318.0
wirc0007.fits.gz     12313.0
wirc0008.fits.gz     12313.0
wirc0009.fits.gz     12309.0
wirc0010.fits.gz     12305.0
wirc0011.fits.gz     12303.0
Dome Off:
wirc0012.fits.gz     52.0000
wirc0013.fits.gz     49.0000
wirc0014.fits.gz     49.0000
wirc0015.fits.gz     48.0000
wirc0016.fits.gz     48.0000
wirc0017.fits.gz     47.0000
wirc0018.fits.gz     47.0000
wirc0019.fits.gz     48.0000
wirc0020.fits.gz     48.0000
wirc0021.fits.gz     47.0000
Computing Flat.
Writing jflat.fits
```

At right is a typical dome flat.

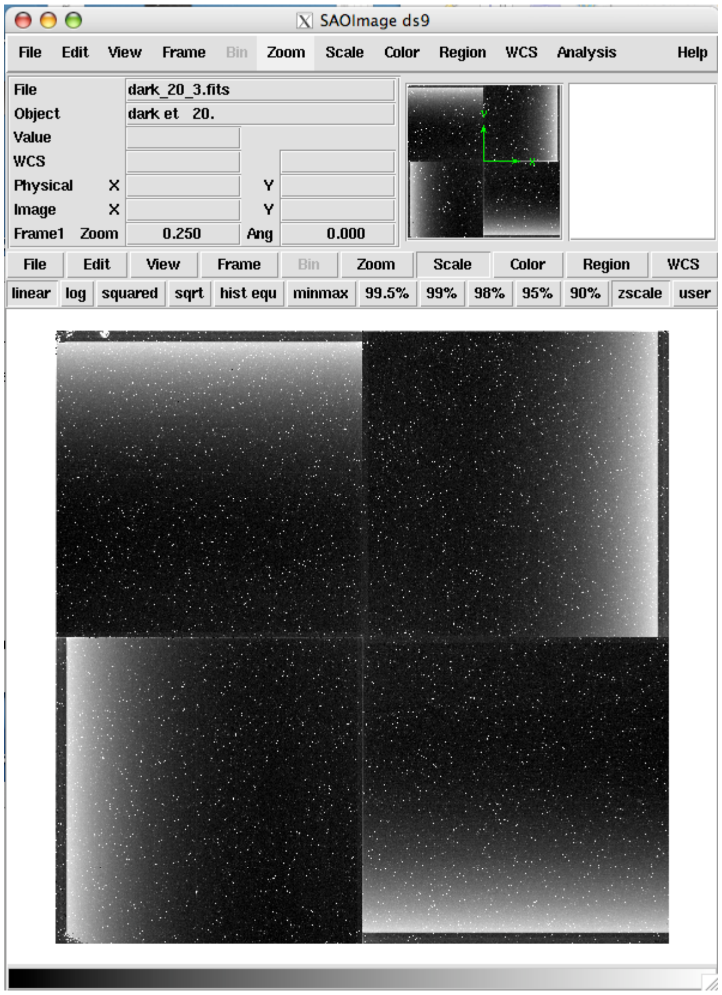

SAOImage ds9
File Edit View Frame Bin Zoom Scale Color Region WCS Analysis Help
File jflat.fits
Object
Value
WCS
Physical X Y
Image X Y
Frame1 Zoom 0.250 Ang 0.000
File Edit View Frame Bin Zoom Scale Color Region WCS
linear log squared sqrt hist equ minmax 99.5% 99% 98% 95% 90% zscale user

3. Make Darks

Here, the two arguments are the logsheet indices as above. Alternately, you could specify any file list and these indices would just be the first and last position in that file (starting at 0). Again, the code automatically selects a filename such that other routines can automatically locate the calibration files. The filename contains the exposure time and the number of coadds. You could also specify your own name (see the task header).

```
IDL> wirc_makedark,'all.list',62,71,/VERBOSE
% READCOL: 397 valid lines read
wirc0062.fits.gz
wirc0063.fits.gz
wirc0064.fits.gz
wirc0065.fits.gz
wirc0066.fits.gz
wirc0067.fits.gz
wirc0068.fits.gz
wirc0069.fits.gz
wirc0070.fits.gz
wirc0071.fits.gz
Writing dark_20_3.fits
```

At right is a typical dark.

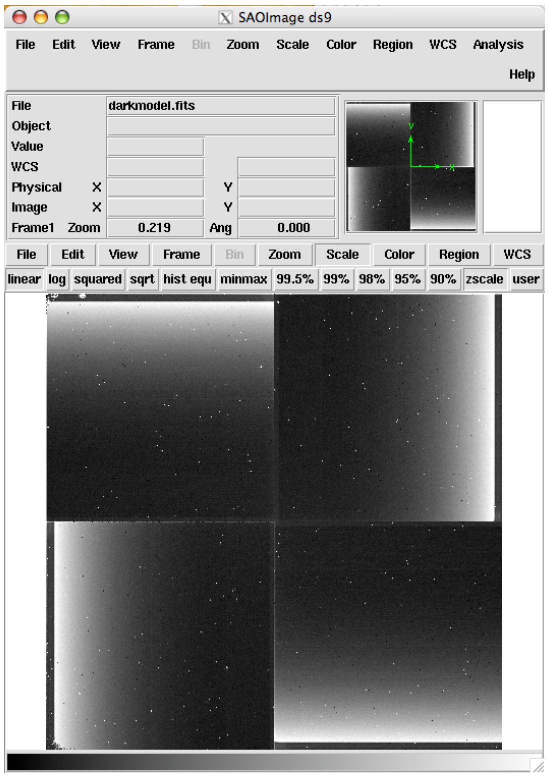

SAOImage ds9
File Edit View Frame Bin Zoom Scale Color Region WCS Analysis Help
File Edit View Frame Bin Zoom Scale Color Region WCS
linear log squared sqrt hist equ minmax 99.5% 99% 98% 95% 90% zscale user

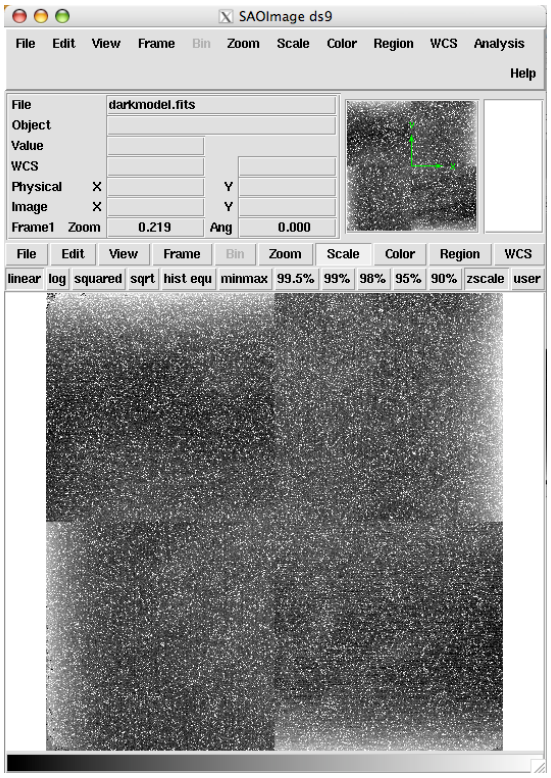
File: dark_20_3.fits
Object: dark et 20.
Value
WCS
Physical X Y
Image X Y
Frame1 Zoom 0.250 Ang 0.000

3b. (or) Make the Darkmodel

The dark current may also be approximated using a model. This will work if you took a bunch of darks with different exposure times and a single coadd. The data are fit as a function of exposure time. The resulting file "darkmodel.fits' contains 2 data planes. The first plane contains the bias pattern, the second the actual dark current in DN/sec. In general, it is better to use darks corresponding to exactly the exposure time/coadd combination you actually used. However, barring that, this model will do in a pinch. If wirc_runningskysub fails to find the appropriate dark automatically, it will search for the dark model. If found, it will reconstruct the appropriate dark from the model. Note that in "fullauto" mode, the task will automatically find all the appropriate darks in the input list, so it does your thinking for you.

```
IDL> wirc_makedarkmodel,'all.list',/fullauto,/verbose
% READCOL: 420 valid lines read
Found          71 out of         420
Reading dark wirc0001.fits.gz
Reading dark wirc0082.fits.gz
…
Reading dark wirc0149.fits.gz
Reading dark wirc0150.fits.gz
Reading dark wirc0151.fits.gz
Fitting dark model.
Writing darkmodel.fits
```


SAOImage ds9
File Edit View Frame Bin Zoom Scale Color Region WCS Analysis Help
File darkmodel.fits
Object
Value
WCS
Physical X Y
Image X Y
Frame1 Zoom 0.219 Ang 0.000
File Edit View Frame Bin Zoom Scale Color Region WCS
linear log squared sqrt hist equ minmax 99.5% 99% 98% 95% 90% zscale user


SAOImage ds9
File Edit View Frame Bin Zoom Scale Color Region WCS Analysis Help
File darkmodel.fits
Object
Value
WCS
Physical X Y
Image X Y
Frame1 Zoom 0.219 Ang 0.000
File Edit View Frame Bin Zoom Scale Color Region WCS
linear log squared sqrt hist equ minmax 99.5% 99% 98% 95% 90% zscale user

4. Make the Object Masks

Now we will make object masks, which are needed to produce the best-quality sky subtraction. This first step also produces sky-subtracted, flat-fielded images, but they will be overwritten on the next step by higher quality versions.

```
IDL>wirc_runningskysub,'dog.list',3,/COMPRESS,RUN=1
```

Many things will happen:

a. Each object frame in turn will be read in.
b. N data frames on each side of the frame being processed will be read in. The file list ('dog.list' above) must be therefore be time-ordered. Also, all frames must be taken with the same number of coadds and exposure time. The frames must also be dithered. These frames are median stacked to form a sky image.
c. Both the object and the sky image are dark subtracted.
d. The sky image is subtracted from the dark. Note that there is a pedestal offset applied to make the result of this come out to a mean of zero.
e. The object is then flat-fielded.
f. A routine is run to try and minimize quadrant bias-offsets.
g. Transfer the world coordinate system based on the telescope TCS to the image.
h. Sextractor is run and a mask image is output.

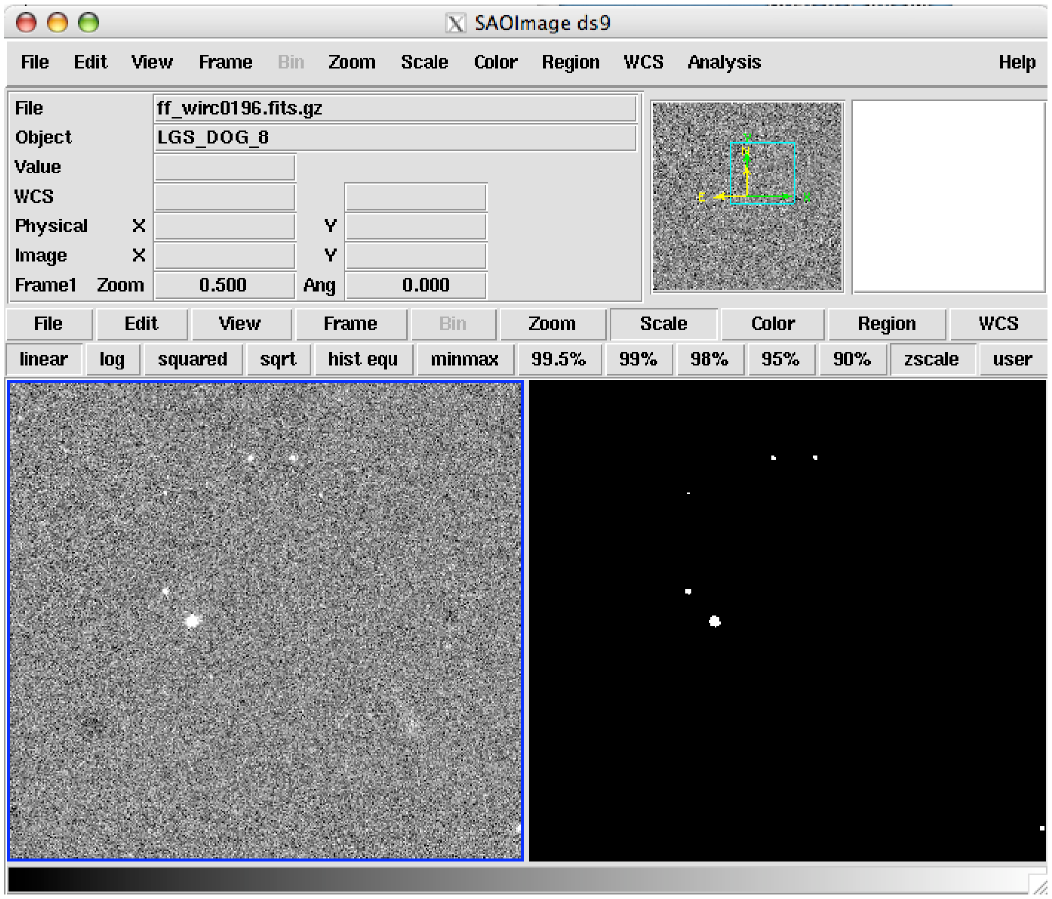
The task will automatically locate the appropriate calibration files, or alternately you can specify them explicitly on the command-line (see the task header). Informational messages will be printed to the screen, the most important of which are the current frame number and the sky files being read in. Speed on a modern (for 2006; 2-core 2.7GHz) computer will be about 30 seconds per image. Note that everything is I/O-heavy, so you will want to run this on a local disk.

In the options shown above, "3" means 3 images on each side of the target frame. For WIRC this is nearly always the right choice. /COMPRESS means to output the files in compressed form (they are called mask_* and ff_*). Both Gaia and DS9 read gzipped files directly, and this saves a lot of space. RUN=1 is important, and means that this is the mask generating step.

If you see sextractor errors, check to make sure that "sex" is in your path, and that you remembered to copy the sex_files directory to your local working directory. Note that there are several hardwired sextractor paths. If your system does not conform to the install structure that mine does, you will need to edit these. The files that need editing are:

Wirc_runningskysub (path to sextractor)
Wirc_fixwcs (path to sextractor)
Sex_files/preproc_mask.sex (path to sextractor support files)
Sex_files/fixwcs.sex (path to sextractor support files)

The result is a sky-subtracted, flat-fielded frame, and an object mask image that corresponds to that frame. Only the latter is actually needed to proceed with the next step, since we are going to repeat all this again. It is possible to also save the sky frames, but in practice there is very little point to doing so.

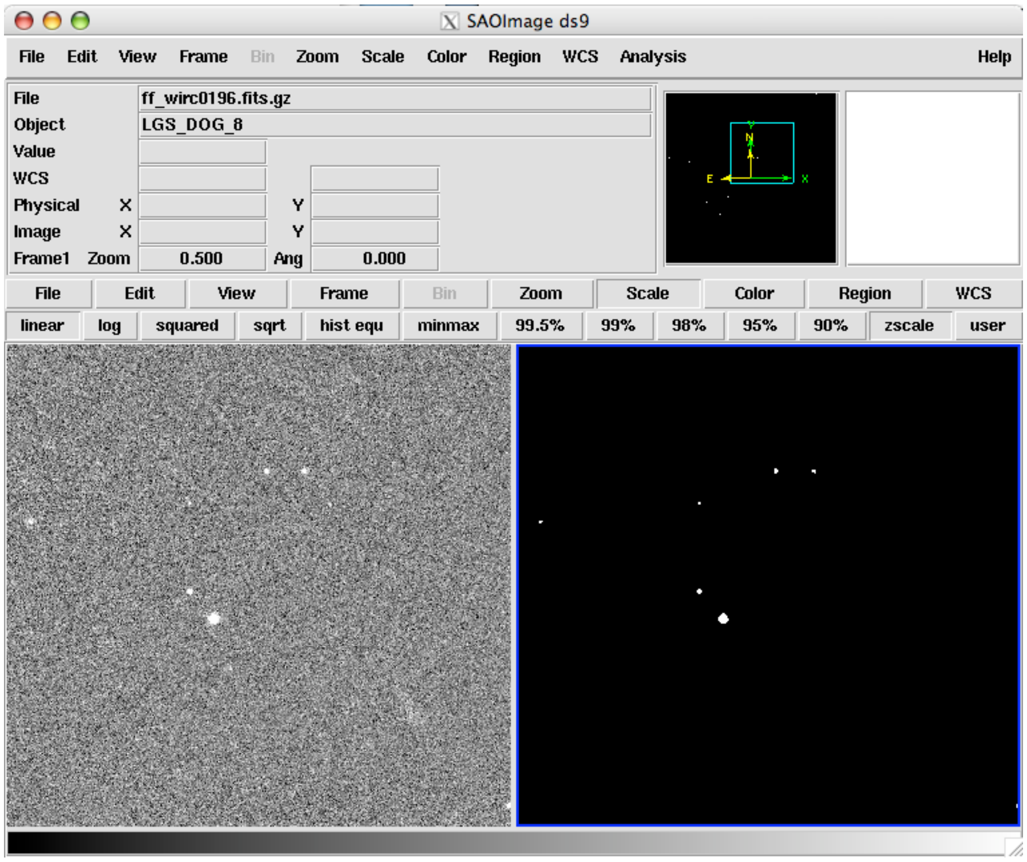

SAOImage ds9
File Edit View Frame Bin Zoom Scale Color Region WCS Analysis Help
File ff_wirc0196.fits.gz
Object LGS_DOG_8
Value
WCS
Physical X Y
Image X Y
Frame1 Zoom 0.500 Ang 0.000
File Edit View Frame Bin Zoom Scale Color Region WCS
linear log squared sqrt hist equ minmax 99.5% 99% 98% 95% 90% zscale user

At left is a first-pass, sky-subtracted, flat-fielded image. Note the residual oversubtractions from objects in the sky images. At right is the corresponding mask image.

4. Make the Flattened, Sky-Subtracted Images

Now we repeat the last command, but in this case, RUN=2 indicates that this is the second pass through the data. Exactly the same steps will be carried out as before, only when the sky frame is made, it will read in the mask files from the previous run and use them to reject pixels that contained objects. The result should be a very high quality sky subtraction. Note that occasionally the quadrant cleaner malfunctions (the final, combined image will have linear vertical and horizontal depressions in it). You have the option of using /NOQUADCLEAN to turn it off.

```
IDL> wirc_runningskysub,'dog.list',3,/COMPRESS,RUN=2
```



File: ff_wirc0196.fits.gz
Object: LGS_DOG_8
Value
WCS
Physical X Y
Image X Y
Frame1 Zoom 0.500 Ang 0.000

5. Get a 2MASS Catalog for the Data.

First, we download a 2MASS catalog, which will be the basis of all the calibration. Please not that the result is just a text file consisting of RA/DEC (in degrees) and J,H,K magnitudes. This task used the query_vizier command in IDLastro. I have had significant problems with this command changing between the astro library revisions, so I cannot guarantee it will work out of the box.

Even if you used a different filter than a 2MASS filter, you can you still use this for the astrometric calibration.

```
humu% ls ff*gz > dog_ff.list
```

```
IDL> wirc_get_2mass,'dog_ff.list','dog_2mass.tbl',/VERBOSE
% READCOL: 67 valid lines read
Using RA,DEC center of       217.606      34.4670 degrees.
Radius is       12.5312 arcminutes.
```

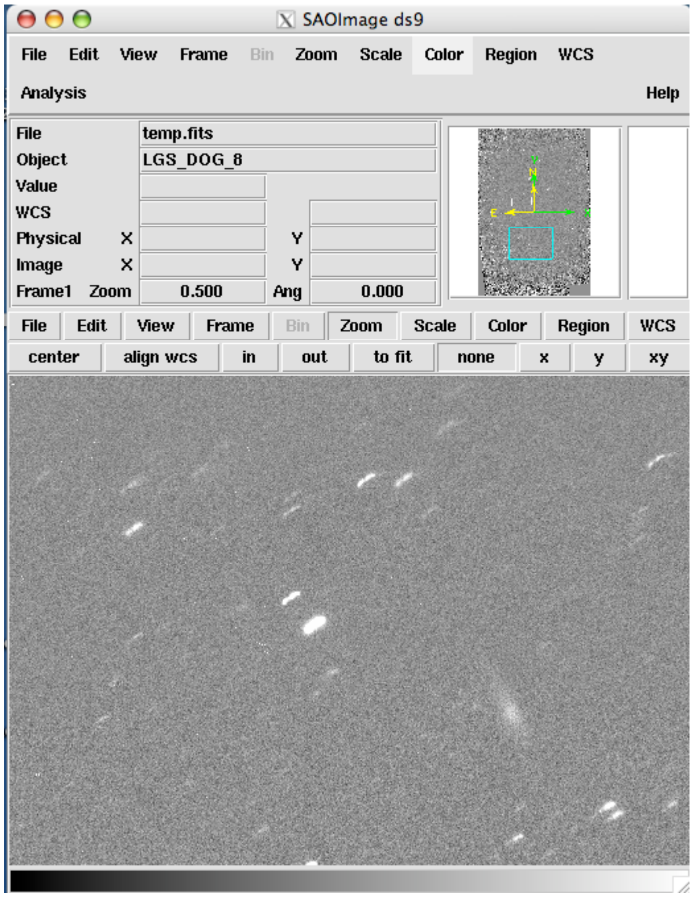
```
humu% more dog_2mass.tbl
      217.54154     34.266895     11.1240     10.6460     10.5800
      217.54044     34.291012     16.4820     15.7570     15.3710
      217.52495     34.310699     15.1410     14.5960     14.5720
      217.51101     34.301083     17.1190     16.0550     16.0000
```

6. Astrometric and Photometric Calibration

In the old days, we had to manually align frames to each other based on whatever was there, and we had to take standard stars and just hope the sky didn't change much during the observations. With 2MASS, that has all changed for wide-angle cameras, as there are high-quality position and flux measurements in every frame. Hence, each and every frame can be individually calibrated.

Why astrometrically calibrate? The Palomar 200 inch telescope has a drift rate of roughly 5-10 arcseconds per hour in pointing. Given how short WIRC images must be, this means that it can easily be used in unguided mode during individual exposures. However, this does mean that for objects observed over many minutes or hours, the coordinates from the TCS are not exact enough to allow for coaddition.

The figure below illustrates a coaddition of flat-fielded, sky-subtracted frames taken over the course of 1.5 hours, and using the telescope TCS pointing to provide the offsets. Drifts in the telescope pointing results in the smearing of the image seen here. As a result, it is necessary to align all the frames.


SAOImage ds9
File Edit View Frame Bin Zoom Scale Color Region WCS
Analysis Help
File Edit View Frame Bin Zoom Scale Color Region WCS
center align wcs in out to fit none x y xy

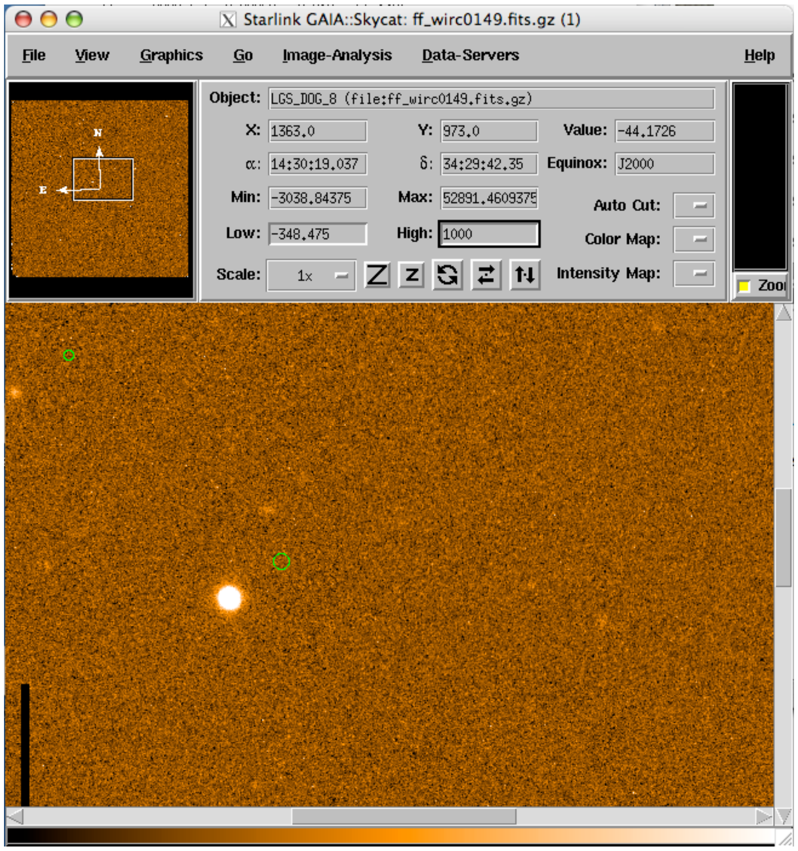
File temp.fits
Object LGS_DOG_8
Value
WCS
Physical X Y
Image X Y
Frame1 Zoom 0.500 Ang 0.000

The next step astrometrically and photometrically calibrates the data versus 2mass. The code will automatically determine which filter (J,H, or Ksh) to calibrate against. If you used a different filter, the code will fall back to K. This will at least let you do the astrometric corrections, even if the photometry is uncalibrated. It is important to realize that the zeropoint written to the headers is the zeropoint for this particular image, in the manner that it was taken. By this, I mean specifically the number of coadds. If the number of coadds changes, then the zeropoint changes by 2.5*log(coadds). It is assumed that all of the observations for a given target are taken with the same exposure time and number of coadds.

The wirc_fixwcs command runs sextractor in order to get a source list of objects in the frame. It then measures the offset between this and the 2mass position list, and computes the mean offset to place the detected objects at the coordinates indicated by 2mass. Note that this implies that the WCS written to the header by the TCS is approximately correct, as the software implicitly assumes that the bright objects in the field match to the closest objects in 2MASS. If the current pointing is very off (say, greater than an arcminute), then you need to manually give the software the offset to the correct position in the first frame. You need only specify this approximately, and for the first frame only. After that the code will "lock" to the correct solution and follow the pointing drift.

The pointing on the 200inch is extremely good. It is rare that this will be needed. In general, you should not bother giving the code offsets (it defaults to [0,0]).

Offsets are provided as a 2-element vector called OFFSETS. This is the x,y value in pixels to move from the actual location to the perceived location of the star. In the example below, we have used GAIA to examine the first frame in this image sequence, and then used GAIA's catalog services to overlay a 2MASS catalog, based on the telescope TCS. The offsets needed are 30 pixels in x and y. Note that for this image you would have been close enough anyway that you could have used the default, which is OFFSETS=[0,0].



Object: LGS_DOG_8 (file:ff_wirc0149.fits.gz)
X: 1363.0 Y: 973.0 Value: -44.1726
α: 14:30:19.037 δ: 34:29:42.35 Equinox: J2000
Min: -3038.84375 Max: 52891.4609375
Low: -348.475 High: 1000
Scale: 1x

```
IDL> wirc_fixwcs,'dog_ff.list','dog_2mass.tbl',OFFSETS=[30,30],/VERBOSE
Comparing vs. 2MASS Ks.
-----------------------------------------------------------------
ff_wirc0149.fits.gz

Ks__(2.15)      20.0000          3
Local Sidereal Time:        12.404111 hours
Number of sextractor detected objects:         144
Median seeing:      1.48304 arcseconds.
Number of potential match stars:          22
RA Median, Mean, Sigma offset (arcsec) of local fit:
     -3.74914     -0.535057      0.161165
DEC Median, Mean, Sigma offset (arcsec) of local fit:
    -0.310599      -1.33885      0.123778
Magnitude zeropoint:       27.0445
Total offsets       -12.802490       7.1503969 arcseconds.
Before offsets (pixels)       30       30
After offsets (pixels)        42       28
-----------------------------------------------------------------
```

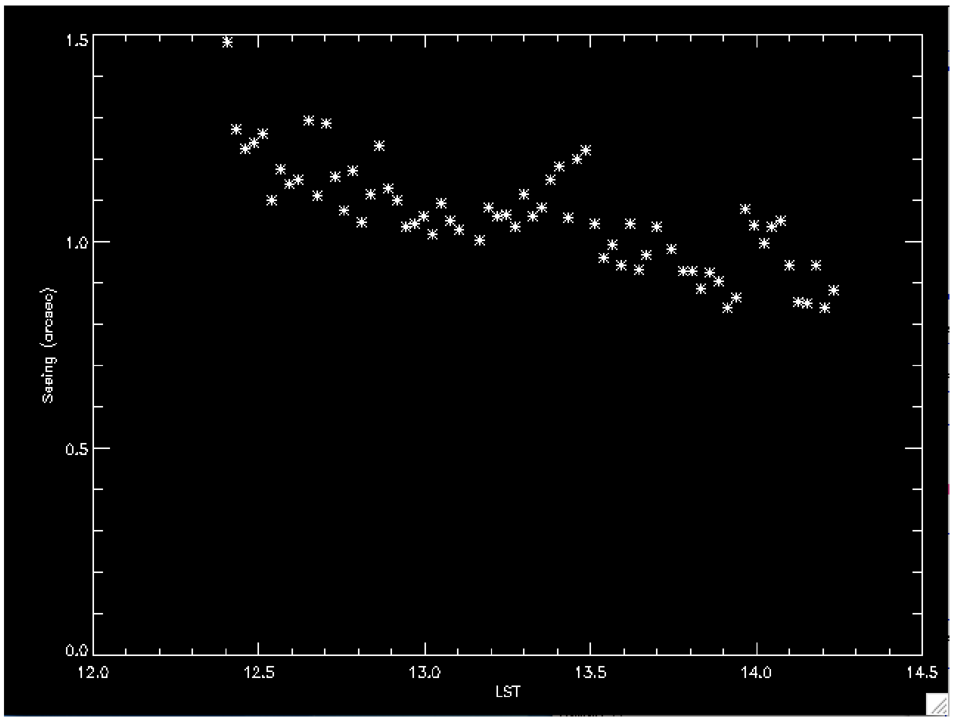
7. Examine Frame-Level Diagnostics

The next command will make a variety of informative plots about the data, which will be displayed to the screen. The optional PLOTROOT keyword will cause the plots to be saved as gif files. You will probably want to note the photometric zeropoint. Also, the scatter in the zeropoint will dictate how you coadd the frames.

```
humu% ls wcs*fits > dog_wcs.list
IDL> wirc_diagnostics,'dog_wcs.list',PLOTROOT='dog'
% READCOL: 67 valid lines read
Loading magnitude zeropoints.
Median magnitude zeropoint is       27.0630
Average is       27.0589 +/-    0.0319357
Plotting median magnitude zeropoint vs. LST.
Press any key to continue...
Plotting mean RA offset vs. LST
Press any key to continue...
Plotting mean DEC offset vs. LST
Press any key to continue...
Plotting mean pointing error vs. LST.
Press any key to continue...
Plotting median seeing vs. LST.
Median seeing was       1.04904 arcseconds.
Average was       1.06085+/-     0.126500 arcseconds.
```

```
humu% ls *gif
dog_decoffset.gif      dog_pnterr.gif        dog_seeing.gif
dog_magzpt.gif         dog_raoffset.gif
```

```
humu% xv dog_seeing.gif
```

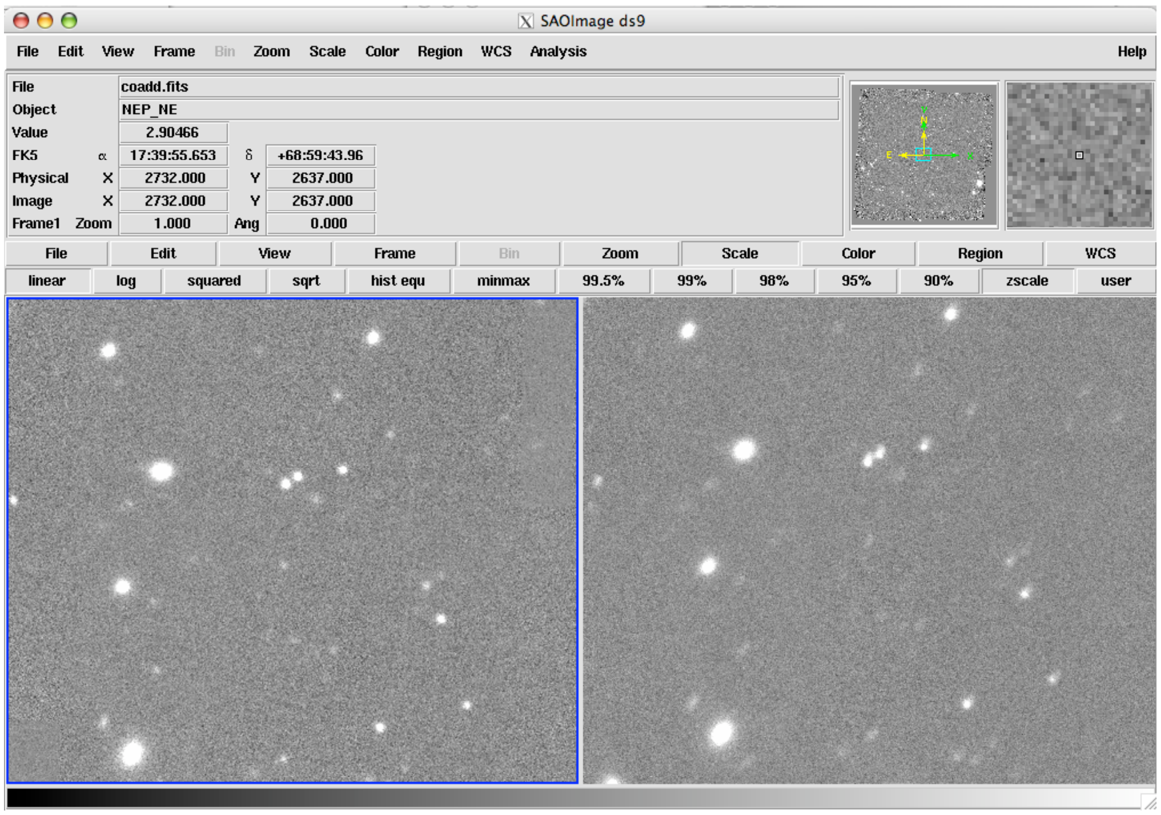


8. Coaddition

Now we reach the hardest part of this whole process. You will want to coadd the images together (for obvious reasons). After some experimentation I have settled on the approach described below. It is a hybrid of SWARP and IRAF.IMCOMBINE. Why do this? There are a couple reasons.

There is a significant issue in terms of coadding many WIRC frames if they cover a large area of sky (say > 2 WIRC FOVs in diameter). Specifically, you will need to reproject the images. In the examples above, the target was relatively small, and the observations consist primarily of a single dithered pointing. However, in the case of large mosaics, you will not be able to just stack the images on top of each other because they will not line up properly at the edges. SWARP will do the reprojections.

Unfortunately, SWARP by itself probably isn't good enough. Why? Although SWARP is both very fast, and does a great job of reprojection, it has extremely limited capability in terms of outlier rejection. IRAF.IMCOMBINE is much more powerful in terms of "knobs" that can be twiddled to improve the outlier rejection.


SAOImage ds9
File Edit View Frame Bin Zoom Scale Color Region WCS Analysis Help
File Edit View Frame Bin Zoom Scale Color Region WCS
linear log squared sqrt hist equ minmax 99.5% 99% 98% 95% 90% zscale user
N
E

```
File          coadd.fits
Object        NEP_NE
Value         2.90466
FK5       α   17:39:55.653    δ  +68:59:43.96
Physical  X   2732.000        Y  2637.000
Image     X   2732.000        Y  2637.000
Frame1 Zoom   1.000         Ang  0.000
```

The left image has been coadded after reprojection with swarp, the one at right has not. The edges don't line up properly.

Hence, we use a hybrid approach. The task "wirc_irafprep" is used to preprocess the images so that they can then be combined in IRAF. It does the following:

a) Reads in some vital information from the headers, specifically the magnitude zeropoint.
b) Writes the keyword IRAFSCL to the header. This is a multiplicative factor needed to place a given image on the same zeropoint as the median of the input zeropoints.
c) Writes the keyword IRAFWGT, which is the inverse of IRAFSCL.
d) Uses SWARP to reproject all images to a common projection center. You must specify /REPROJECT for this to happen.
e) Replaces all NaNs and zero coverage areas with an absurdly negative number (-9999). This is required because IRAF translates NaNs to zero, which is exactly the wrong thing to do.
f) Also zaps all pixels that have low sensitivity in the flatfield. You can override this.
g) You can also have SWARP produce it's own coadd by specifying the filename with SWARPFILE. However, if your depth of coverage is very high, this will die with memory problems. For WIRC it rarely works.

```
IDL> wirc_irafprep,'dog_wcs.list',/REPROJECT
```

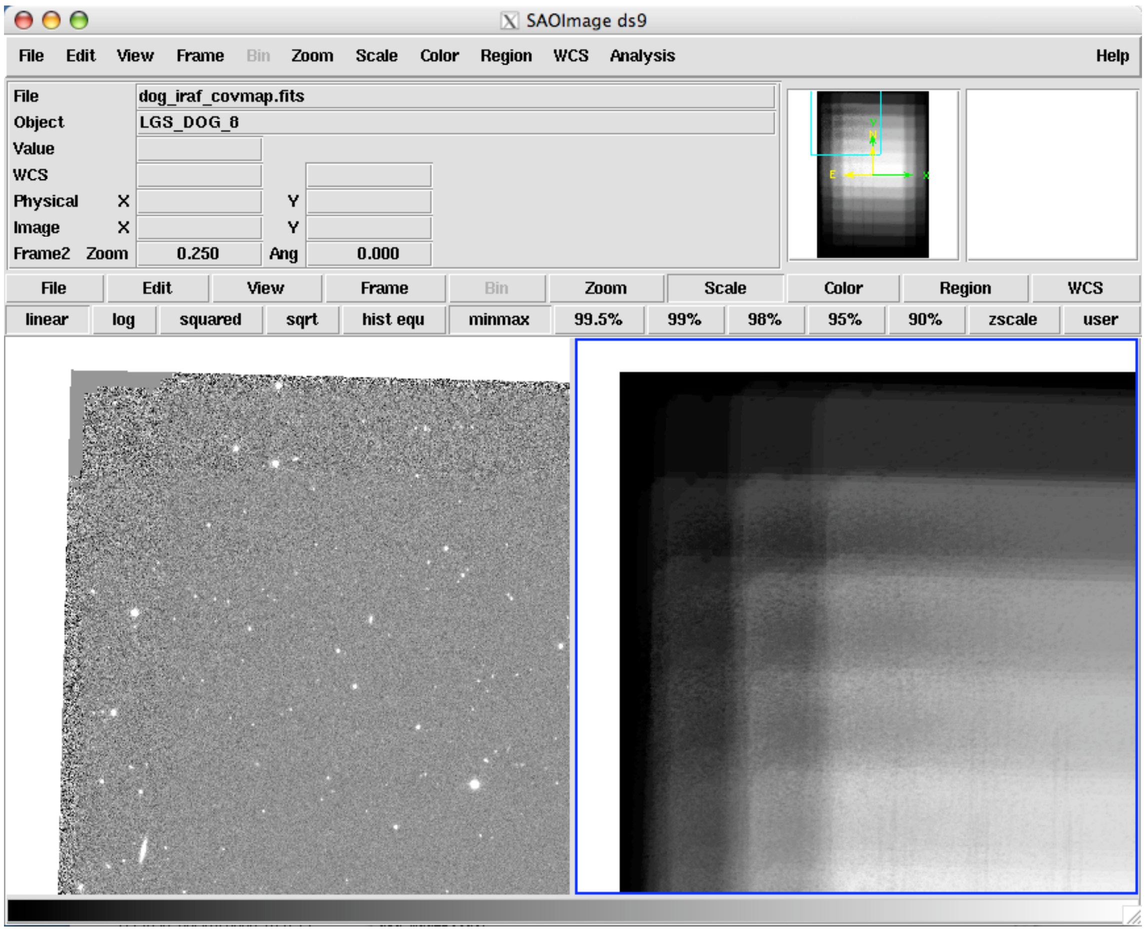
Once this is done, the images (prep_*) can then be coadded in IRAF. Specific psets are shown in the appendix. The key items are to set the offsets=WCS, and the lower data limit to -900. You can use whatever outlier rejection you wish. Should you use IRAFSCL and IRAFWGT? This really depends on the conditions. If you are coadding data from nights with changing conditions, then yes. If the night was very photometric, then I wouldn't bother. There are several important caveats to this step.

First, SWARP removes a local background from the images. You won't need to set a background offset in IRAF. But it does mean that complex backgrounds could get erased.

Second, SWARP normally chooses a field center based on the input data. This may not be desirable if, for example, you have multiple filters. In that case you would want them all to be on the same projection center. You should therefore edit the sex_files/wirc_irafprep.swarp file to manually insert the pointing center.

Third, the IRAFSCL parameters are written relative to the median of the input stack. This is a problem if you later expect to coadd different days which were processed separately. In this case, use FIXZP=25 so that the IRAFSCL keywords are set relative to a manually input value (in this case 25). Note that a peculiarity of IRAF is that the output image scale is set equal to that of the first input image. So even if you set 25 as the zeropoint, the final coadd will default to the first image zeropoint.

Finally, there is a /COVMAP option. If this is used, the output pixels are set equal to "1", such that when combined with IRAF using the "sum" option you will get an image which is the depth at each point in terms of number of images taken. This will be useful later if you wish to use Sextractor or similar software.


SAOImage ds9
File Edit View Frame Bin Zoom Scale Color Region WCS Analysis Help
File Edit View Frame Bin Zoom Scale Color Region WCS
linear log squared sqrt hist equ minmax 99.5% 99% 98% 95% 90% zscale user

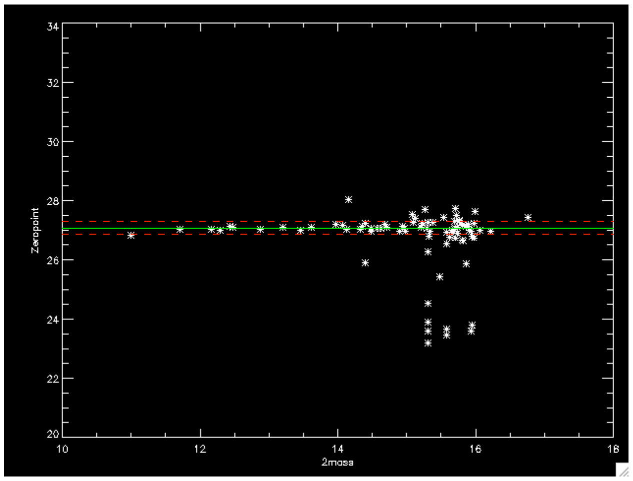
File: dog_iraf_covmap.fits
Object: LGS_DOG_8
Value
WCS
Physical X Y
Image X Y
Frame2 Zoom 0.250 Ang 0.000

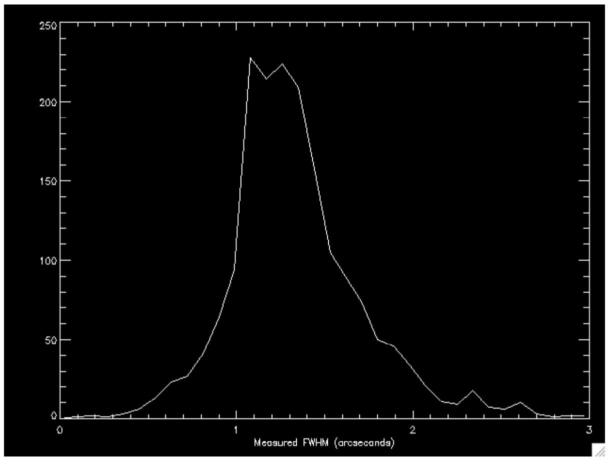
The above shows a coadded image at left, and the coverage map at right.

9. Final Calibration

Now that the data is coadded and mosaiced, you will want to check the final calibration. In principle, this step is not needed, as the individual frames were forced onto the 2MASS astrometric and photometric calibration. However, it would be wise to check. In particular, it is fairly common to get small (1-2 pixel) offsets in the astrometry. Similarly, the exact photometric calibration depends to some extent on how the frames were weighted when coadded, and this is particularly so on non-photometric nights. With the following command, sextractor will be run and the results compared to 2MASS. A final image will be written, called in this case "dog_iraf_swarp_finalcalib.fits". The astrometry will have been updated, and a new magnitude zeropoint will be written.

```
IDL> wirc_finalcalib,'dog_iraf_swarp.fits',TMASSFILE='dog_2mass.tbl'
MEANZPT =          27.0768 / mean magnitude zeropoint
MEDZPT  =          27.0752 / median magnitude zeropoint
SIGMAMAG=          0.0745836 / 1-sigma scatter in zeropoint
MEDSEE  =          1.33552 / median observed FWHM (arcsec)
Q1SEE   =          1.14651 / first quartile observed FWHM (arcsec)
```





Note – there is an ambiguity in the astrometry that arises because Sextractor and IDLastro disagree on the definition of the first pixel. Sextractor uses the convention that the first pixel is located at (1,1), whereas the IDL routines define it to be (0,0).

Sample parameter set for making a mosaic from wirc_irafprep data:

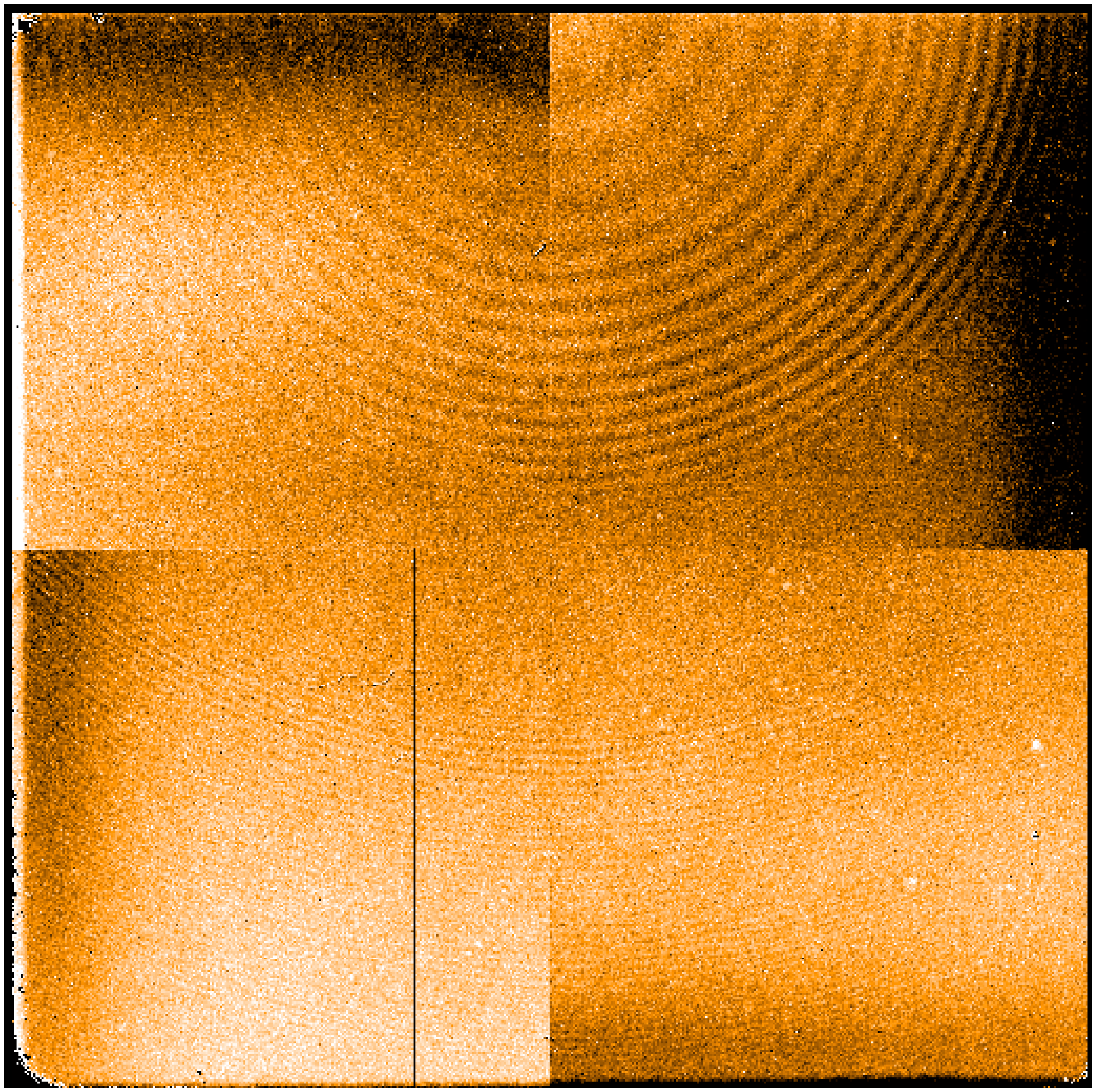
```
ecl> lpar imcomb
        input = "prep*fits"     List of images to combine
       output = "temp2.fits"    List of output images
     (headers = " ")            List of header files (optional)
     (bpmasks = " ")            List of bad pixel masks (optional)
    (rejmasks = " ")            List of rejection masks (optional)
   (nrejmasks = "")             List of number rejected masks (optional)
    (expmasks = "")             List of exposure masks (optional)
      (sigmas = "")             List of sigma images (optional)
     (logfile = "STDOUT")       Log file\n
     (combine = "average")      Type of combine operation
      (reject = "avsigclip")    Type of rejection
     (project = no)             Project highest dimension of input images?
     (outtype = "real")         Output image pixel datatype
   (outlimits = "")             Output limits (x1 x2 y1 y2 ...)
     (offsets = "wcs")          Input image offsets
    (masktype = "none")         Mask type
   (maskvalue = "0")            Mask value
       (blank = 0.)             Value if there are no pixels\n
       (scale = "!IRAFSCL")     Image scaling
        (zero = "median")       Image zero point offset
      (weight = "!IRAFWGT")     Image weights
     (statsec = "[300:700,300:700]") Image section for computing statistics
     (expname = "")             Image header exposure time keyword\n
  (lthreshold = -900.)          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)           pclip: Percentile clipping parameter
        (grow = 0.)             Radius (pixels) for neighbor rejection
        (mode = "ql")
```

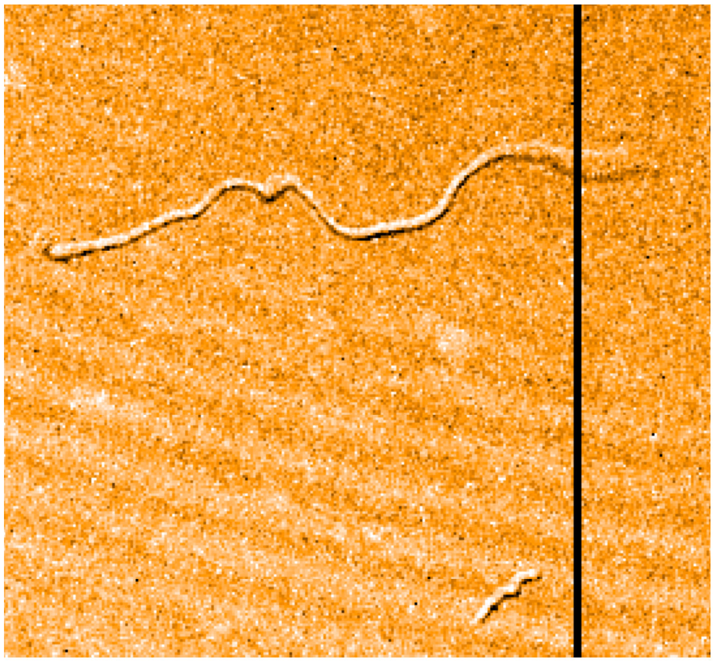
Sample parameter set for making a coverage map from wirc_irafprep data:

```
input = "prep*fits"     List of images to combine
       output = "temp6.fits"    List of output images
     (headers = " ")            List of header files (optional)
     (bpmasks = " ")            List of bad pixel masks (optional)
    (rejmasks = " ")            List of rejection masks (optional)
   (nrejmasks = "")             List of number rejected masks (optional)
    (expmasks = "")             List of exposure masks (optional)
      (sigmas = "")             List of sigma images (optional)
     (logfile = "STDOUT")       Log file\n
     (combine = "sum")          Type of combine operation
      (reject = "none")         Type of rejection
     (project = no)             Project highest dimension of input images?
     (outtype = "real")         Output image pixel datatype
   (outlimits = "")             Output limits (x1 x2 y1 y2 ...)
     (offsets = "wcs")          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 = " ")            Image weights
     (statsec = " ")            Image section for computing statistics
     (expname = "")             Image header exposure time keyword\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)           pclip: Percentile clipping parameter
        (grow = 0.)             Radius (pixels) for neighbor rejection
        (mode = "ql")
```

Some Notes about the Use of Skyflats vs Dome Flats

It is clear that the skyflats are illuminated slightly differently from the dome flats. Shown below is the ratio between dome flats and night sky flats.





The main features are a clear interference pattern, and differences in the vignetting on the left side and the illumination of dust particles and fibers. Overall, the use of the dome flats injects about 0.5% additional noise into the images. At worst, the errors are typically of order 2%. Note that when using sky-subtraction, these errors are coherent and subtract out, so it is almost impossible to tell, looking at the images, that an error has arisen. However, the errors do exist.