ARCTIC Photometry Data Reduction Cookbook

Karen Kinemuchi
Apache Point Observatory, Sunspot, NM

Version 2.1 - February 2018

DISCLAIMER: I have made this write up to reduce my quad-mode data from the ARCTIC photometer using IRAF/PyRAF. It is assumed the user of this cookbook is familiar with the IRAF environment and the basic steps of data reduction for CCD images. If not, please refer to the excellent manuals provided by the folks at IRAF[1]. My nomenclature for many of the input and output files were made to keep things simple and less confusing for me, and my choices may not be the best for you. Caveat Emptor.

Please visually inspect your data throughout the processing to make sure you are getting things that are sensible. If not, step back and check if the previous task did not screw up or introduce something wrong.

To use this cookbook, all IRAF tasks are in ALL CAPS for readability. I also provide the parameters for many of the IRAF tasks for convenience, but the user is encouraged to play around with some of the parameters to optimize the steps for their individual science. Keep in mind, I do variable stars, so I have optimized the parameters for my science.

ACKNOWLEDGEMENTS: The IRAF cl scripts that actually perform the trimming and stitching of the quad mode images are Phil Massey's Y4K Camera reduction scripts. I have only modified the overscan trim sections and renamed the scripts for ARCTIC data reduction purposes. Interested readers can find more information about these scripts or can download the original from his website[2].

Also a big thanks to Jack Dembicky of APO for a careful read through and test driving of the modified Y4K Cam scripts.

If you have any questions or suggestions, feel free to contact me: kinemuchi@apo.nmsu.edu

[1] http://iraf.noao.edu/docs/spectra.html – *A User's Guide to CCD Reductions with IRAF* and *A Beginner's Guide to Using IRAF*

[2] http://www2.lowell.edu/users/massey/obins/y4kcamred.html

INITIAL SETUP

To get started, you will need to load the following IRAF packages:

IMRED
CCDRED
BIAS

Also have a ds9 or saoimage window up to visually inspect all your data. Before starting this processing, PLEASE look your calibration images (biases, darks, and flat fields). Do they look ok? If there are any weird artifacts or abnormal issues, contact the observation specialists at the 3.5m, and they may be able to tell you if the instrumental effects are real in your data.

1) To start, modify your `login.cl` file to directly read in the new IRAF tasks. Once you place these lines in your `login.cl`, you can skip this step for future data processing.

Specifically, add these lines to the User Package Section in your login.cl file:

```
# For quad mode imaging data
task    apobreak=~/YOUR_IRAF_DIR/apobreak.cl
task    apotproc=~/YOUR_IRAF_DIR/apotproc.cl
task    aponew=~/YOUR_IRAF_DIR/aponew.cl
```

Where "YOUR_IRAF_DIR" will depend on where you keep your login.cl file – mostly your /iraf directory. Be sure that the cl scripts are in this directory too.

If you start up IRAF, then these tasks will be automatically loaded. To check if the task is active, try typing in IRAF `lpar apobreak`.

HOWEVER: If you cannot open up the edit parameters window (epar) for any of these tasks, then try manually loading the tasks using the following commands in IRAF, typed at the cl prompt:

```
task apobreak = apobreak.cl
task apotproc = apotproc.cl
task aponew = aponew.cl
```

Input these commands in the directory where the cl source code is located (most likely your iraf directory). You should be able to cd to your data directory where you want to do your processing, and these tasks still should work (check by doing an lpar of these tasks - type at the cl prompt `lpar apobreak`).

TRIMMING and STITCHING QUAD IMAGES

2) The first thing to do is to break up the single image into 4 smaller images corresponding to each quadrant. The IRAF task to use here is APOBREAK. You can run these commands for each individual file (if you only have a few), or write a cl script to process all of the images. For example, here is an example of a simple cl script file you can create to just process the bias images. The cl script is called *dobias.cl*. You can create additional scripts using the same format for the rest of your images (darks, flats, object images) or just use the same one, but with different filename.

I have provided in the Appendix the full *dobias.cl* script used to run the entire processing. In this step of the manual, I'm just going step-by-step with the processing. It is your choice to do this via each step or all at one go.

```
apobreak bias.0011.fits
apobreak bias.0012.fits
apobreak bias.0013.fits
apobreak bias.0014.fits
apobreak bias.0015.fits
```

To run your cl script, simply type at your IRAF cl prompt:

```
cl < dobias.cl
```

In the task APOBREAK, the overscan regions for the quads are defined. The overscan regions are hard coded in the source code for this task. If you find you need to trim more, modify the source code and retask it in IRAF to enact the change.

If you are curious, the overscan regions for the 4 quads are:

```
A: [3:1051,1:1024]  (lower left)
B: [3:1051,1027:2050] (upper left)
C: [1052:2100,1:1024]  (lower right)
D: [1052:2100,1027:2050] (upper right)
```

3) After you break a single image to 4 separate images, the next task to run is APOTPROC. This task uses COLBIAS in the BIAS package. What this task does is apply an overscan correction and bias subtract of the images.

As before, you can apply this task individually to all your images, or just write a cl script to do all of them automatically. For example, in my *dobias.cl* script, I just changed the task name to "apotproc" and ran this script at the cl prompt:

```
apotproc bias.0011.fits
apotproc bias.0012.fits
apotproc bias.0013.fits
apotproc bias.0014.fits
apotproc bias.0015.fits
```

Note: The input file for this task is the original file name, not the intermediate file that was created in the previous task.

You may see a bunch of output that says:

```
Warning: Attempt to delete a nonexistent file (TAobj.0034.fits)
Warning: Attempt to delete a nonexistent file (TBobj.0034.fits)
Warning: Attempt to delete a nonexistent file (TCobj.0034.fits)
Warning: Attempt to delete a nonexistent file (TDobj.0034.fits)
```

You can safely ignore this warning message. The task is simply looking for a previously name file called TA*.fits (or the others) so that you can run this task multiple times.

4) After you have removed the overscan, you want to stitch the four individual images back to the original single image. This stitched single image will be missing the gaps between the quads you see in the raw data. To stitch the 4 subimages back together, use the task APONEW.

As before, you can run this task individually for all your images, or you can write a batch script to do the stitching non-interactively. For example, in my *dobias.cl* script, I just changed the task name to "aponew" and ran this script at the cl prompt:

```
aponew bias.0011.fits
aponew bias.0012.fits
aponew bias.0013.fits
aponew bias.0014.fits
aponew bias.0015.fits
```

Note: The input file for this task is the original file name, not the intermediate files created in the previous task.

If you don't want to interactively delete the intermediate files the script produces, please edit the parameter file for IMDELTE with the parameter set with `verify=no`. Otherwise, you will have to explicitly tell IRAF to "yes" IMDEL Tobj.###.fits for however many files you are processing. Later you can unilaterally delete those intermediate files after the script completes.

Once all three tasks are complete, you will have a bias subtracted, overscan trimmed images with the prefix "T". For example, bias.0001.fits has the corresponding processed file called Tbias.0001.fits. An example of the before and after bias images are shown in Figure 1.

5) Now REPEAT steps 2-4 for your darks (if you have any), for each filter flat, and your object frames. If you have a lot of images, just write a cl script that calls all three tasks and run it in batch mode inside of IRAF/PyRAF. This is especially true if you have multiple filter flats from your observing program. Figure 2 shows the raw and quad-processed V flat field.

DATA REDUCTION STEPS FOR QUAD-MODE DATA

6) After producing the overscan trimmed and stitched images, you are ready to proceed with the usual data reduction steps. Here the procedures are based in IRAF tasks, and I provide the IRAF parameters of each step.

CREATE A MASTER BIAS

6A) Create a master bias using ZEROCOMBINE. My input file (called inmkbias) was created using the IRAF task FILES. The file "inmkbias" becomes your "@-file", in the IRAF parlance.

NB: I have included in the parameter file here with the posted values of the read noise and gain of the ARCTIC chip. The gain is 2.0 electrons/DN and the read noise is 3.7 electrons. Enter in your input @file, the readnoise, and gain.

```
CL> files Tbias* > inmkbias

CL> epar zerocombine
        input = "@inmkbias"     List of zero level images to combine
       (output = "Zero")        Output zero level name
      (combine = "median")      Type of combine operation
       (reject = "avsigclip")   Type of rejection
      (ccdtype = "")            CCD image type to combine
      (process = no)            Process images before combining?
       (delete = no)            Delete input images after combining?
      (clobber = no)            Clobber existing output image?
        (scale = "none")        Image scaling
      (statsec = "")            Image section for computing statistics
         (nlow = 0)             minmax: Number of low pixels to reject
        (nhigh = 1)             minmax: Number of high pixels to reject
        (nkeep = 1)             Minimum to keep (pos) or maximum to reject (neg)
        (mclip = yes)           Use median in sigma clipping algorithms?
       (lsigma = 3.0)           Lower sigma clipping factor
```

```
   (hsigma = 3.0)           Upper sigma clipping factor
  (rdnoise = "3.7")         ccdclip: CCD readout noise (electrons)
     (gain = "2.0")         ccdclip: CCD gain (electrons/DN)
   (snoise = "0.")          ccdclip: Sensitivity noise (fraction)
    (pclip = -0.5)          pclip: Percentile clipping parameter
    (blank = 0.0)           Value if there are no pixels
     (mode = "ql")
```

The master bias frame is called Zero. Check your mean value across the chip by running IMSTAT. Are your mean values around 0?

```
CL>  imstat Zero
#               IMAGE      NPIX      MEAN    STDDEV       MIN       MAX
                 Zero   4194304   -0.1485     10.39    -17.14    18274.
```

CREATE A MASTER DARK

7) Bias subtract all your darks with CCDPROC. The input file, inprocdark, contains all the individual dark frames you want to use. I created this @-file using FILES. To preserve the original input files, I create output files for each bias-subtracted dark file with a prefix "ZTdark*.fits". The simplest way to do this is to make a copy the inprocdark file and then edit the filenames with the new prefix and save the modified input file as "outprocdark".

```
epar ccdproc
       images = "@inprocdark"   List of CCD images to correct
      (output = "@outprocdark") List of output CCD images
     (ccdtype = "")             CCD image type to correct
   (max_cache = 0)              Maximum image caching memory (in Mbytes)
      (noproc = no)             List processing steps only?

      (fixpix = no)             Fix bad CCD lines and columns?
    (overscan = no)             Apply overscan strip correction?
       (trim = no)              Trim the image?
     (zerocor = yes)            Apply zero level correction?
     (darkcor = no)             Apply dark count correction?
     (flatcor = no)             Apply flat field correction?
    (illumcor = no)             Apply illumination correction?
   (fringecor = no)             Apply fringe correction?
     (readcor = no)             Convert zero level image to readout correction?
     (scancor = no)             Convert flat field image to scan correction?
```

```
   (readaxis = "line")         Read out axis (column|line)
    (fixfile = "")             File describing the bad lines and columns
    (biassec = "")             Overscan strip image section
    (trimsec = "")             Trim data section
       (zero = "Zero")         Zero level calibration image
       (dark = "")             Dark count calibration image
       (flat = "")             Flat field images
      (illum = "")             Illumination correction images
     (fringe = "")             Fringe correction images
 (minreplace = 1.0)            Minimum flat field value
   (scantype = "shortscan")    Scan type (shortscan|longscan)
      (nscan = 1)              Number of short scan lines

(interactive = no)             Fit overscan interactively?
   (function = "legendre")     Fitting function
      (order = 3)              Number of polynomial terms or spline pieces
     (sample = "*")            Sample points to fit
   (naverage = 1)              Number of sample points to combine
   (niterate = 3)              Number of rejection iterations
 (low_reject = 3.0)            Low sigma rejection factor
(high_reject = 3.0)            High sigma rejection factor
       (grow = 0.0)            Rejection growing radius
```

7A) Then create a master dark using DARKCOMBINE (or IMCOMBINE). Again the input file is created using FILES. The master dark or "output" is called Dark.

```
epar darkcombine
       input = "@inmkdark"     List of dark images to combine
     (output = "Dark")         Output dark image root name
    (combine = "average")      Type of combine operation
     (reject = "minmax")       Type of rejection
    (ccdtype = "")             CCD image type to combine
    (process = yes)            Process images before combining?
     (delete = no)             Delete input images after combining?
    (clobber = no)             Clobber existing output image?
      (scale = "exposure")     Image scaling
    (statsec = "")             Image section for computing statistics
       (nlow = 0)              minmax: Number of low pixels to reject
      (nhigh = 1)              minmax: Number of high pixels to reject
      (nkeep = 1)              Minimum to keep (pos) or maximum to reject (neg)
      (mclip = yes)            Use median in sigma clipping algorithms?
```

```
      (lsigma = 3.0)           Lower sigma clipping factor
      (hsigma = 3.0)           Upper sigma clipping factor
     (rdnoise = "3.7")         ccdclip: CCD readout noise (electrons)
        (gain = "2.0")         ccdclip: CCD gain (electrons/DN)
      (snoise = "0.")          ccdclip: Sensitivity noise (fraction)
       (pclip = -0.5)          pclip: Percentile clipping parameter
       (blank = 0.0)           Value if there are no pixels
        (mode = "al")
```

Check the mean values of the master dark using IMSTAT. The value should be higher than what you see in the master bias, since the dark is measuring your thermal noise.

```
  imstat Dark
#               IMAGE      NPIX      MEAN    STDDEV       MIN       MAX
                Dark    4194304    0.1335     10.38   -18254.     972.7
```

CREATE MASTER FLAT FIELDS, PER FILTER

8) Now we need to create master flat field frames, per filter. Bias- and dark-subtract all your flat field frames. I use CCDPROC to perform these subtractions. For example, the input @-file, *inprocflat* contains the files "Tflat.*.fits". For the processed output files, I edit the @-file so that each file has a new prefix of "TZDflat*.fits", which in my nomenclature means the file has been Trimmed, Zero subtracted, and Dark subtracted. Makes sense?

```
epar ccdproc
       images = "@inprocflat"  List of CCD images to correct
      (output = "@outprocflat") List of output CCD images
     (ccdtype = "")             CCD image type to correct
   (max_cache = 0)              Maximum image caching memory (in Mbytes)
      (noproc = no)             List processing steps only?

      (fixpix = no)             Fix bad CCD lines and columns?
    (overscan = no)             Apply overscan strip correction?
        (trim = no)             Trim the image?
     (zerocor = yes)            Apply zero level correction?
     (darkcor = yes)            Apply dark count correction?
     (flatcor = no)             Apply flat field correction?
    (illumcor = no)             Apply illumination correction?
   (fringecor = no)             Apply fringe correction?
     (readcor = no)             Convert zero level image to readout correction?
     (scancor = no)             Convert flat field image to scan correction?
```

```
    (readaxis = "line")         Read out axis (column|line)
     (fixfile = "")             File describing the bad lines and columns
     (biassec = "")             Overscan strip image section
     (trimsec = "")             Trim data section
        (zero = "Zero")         Zero level calibration image
        (dark = "Dark")         Dark count calibration image
        (flat = "")             Flat field images
       (illum = "")             Illumination correction images
      (fringe = "")             Fringe correction images
  (minreplace = 1.0)            Minimum flat field value
    (scantype = "shortscan")    Scan type (shortscan|longscan)
       (nscan = 1)              Number of short scan lines

 (interactive = no)             Fit overscan interactively?
    (function = "legendre")     Fitting function
       (order = 3)              Number of polynomial terms or spline pieces
      (sample = "*")            Sample points to fit
    (naverage = 1)              Number of sample points to combine
    (niterate = 3)              Number of rejection iterations
  (low_reject = 3.0)            Low sigma rejection factor
 (high_reject = 3.0)            High sigma rejection factor
        (grow = 0.0)            Rejection growing radius
        (mode = "ql")
```

8A) Now I use FLATCOMBINE to create the master flat, PER FILTER. For example, if you took flat fields in the B, V, and I filters, you should have three master flat fields called "masterbflat", "mastervflat", and "masteriflat". The input @file you enter into FLATCOMBINE will only contain the B TZD*bflat*.fits files, if you are creating a master B flat field, for example. In the example below, I'm creating a master V flat.

```
epar flatcombine
        input = "@invflat"      List of flat field images to combine
      (output = "mastervflat")  Output flat field root name
     (combine = "median")       Type of combine operation
      (reject = "minmax")       Type of rejection
     (ccdtype = "")             CCD image type to combine
     (process = no)             Process images before combining?
     (subsets = no)             Combine images by subset parameter?
      (delete = no)             Delete input images after combining?
     (clobber = no)             Clobber existing output image?
```

```
      (scale = "mode")         Image scaling
    (statsec = "")             Image section for computing statistics
       (nlow = 1)              minmax: Number of low pixels to reject
      (nhigh = 1)              minmax: Number of high pixels to reject
      (nkeep = 1)              Minimum to keep (pos) or maximum to reject (neg)
      (mclip = yes)            Use median in sigma clipping algorithms?
     (lsigma = 3.0)            Lower sigma clipping factor
     (hsigma = 3.0)            Upper sigma clipping factor
    (rdnoise = "3.7")          ccdclip: CCD readout noise (electrons)
       (gain = "2.0")          ccdclip: CCD gain (electrons/DN)
     (snoise = "0.")           ccdclip: Sensitivity noise (fraction)
      (pclip = -0.5)           pclip: Percentile clipping parameter
      (blank = 1.0)            Value if there are no pixels
       (mode = "ql")
```

8B) (optional) I also like to normalize the master flat to 1.0 using IMARITH. To figure out what value you must divide your master flat fields with, run IMSTAT on the master flat field you just created. Use the value under "mean". Here are steps I took to normalize my master V flat:

```
  imstat mastervflat
#               IMAGE      NPIX      MEAN    STDDEV       MIN       MAX
         mastervflat   4194304    17617.     5443.    -232.4    23208.

  imarith mastervflat / 17617.0 normVflat

  imstat normVflat
#               IMAGE      NPIX      MEAN    STDDEV       MIN       MAX
           normVflat   4194304        1.     0.309  -0.01319     1.317
```

FINALLY, CCDPROC

9) Now reduce the object files, per filter. I list here the parameters of CCDPROC for the object files. Again, the input @-file was created with FILES. The output @-file was modified by changing the prefix of the input FITS files. I usually like to use "P" to stand for "processed" for my output files.

Remember to run CCDPROC per filter of your object frames. As in the previous scenario, if you have data taken in the B, V, and I filters, you must identify which of your object frames were taken with which filter and create the input and output files accordingly. Be sure to also change which normalize, master flat field you are using to process your object frames! In the epar below, I am processing my V-band data:

```
epar ccdproc
      images = "@inVobj"       List of CCD images to correct
     (output = "@outVobj")     List of output CCD images
    (ccdtype = "")             CCD image type to correct
  (max_cache = 0)              Maximum image caching memory (in Mbytes)
     (noproc = no)             List processing steps only?

     (fixpix = no)             Fix bad CCD lines and columns?
   (overscan = no)             Apply overscan strip correction?
       (trim = no)             Trim the image?
    (zerocor = yes)            Apply zero level correction?
    (darkcor = yes)            Apply dark count correction?
    (flatcor = yes)            Apply flat field correction?
   (illumcor = no)             Apply illumination correction?
  (fringecor = no)             Apply fringe correction?
    (readcor = no)             Convert zero level image to readout correction?
    (scancor = no)             Convert flat field image to scan correction?

   (readaxis = "line")         Read out axis (column|line)
    (fixfile = "")             File describing the bad lines and columns
    (biassec = "")             Overscan strip image section
    (trimsec = "")             Trim data section
       (zero = "Zero")         Zero level calibration image
       (dark = "Dark")         Dark count calibration image
       (flat = "normVflat")    Flat field images
      (illum = "")             Illumination correction images
     (fringe = "")             Fringe correction images
 (minreplace = 1.0)            Minimum flat field value
   (scantype = "shortscan")    Scan type (shortscan|longscan)
      (nscan = 1)              Number of short scan lines

(interactive = no)             Fit overscan interactively?
   (function = "legendre")     Fitting function
      (order = 3)              Number of polynomial terms or spline pieces
     (sample = "*")            Sample points to fit
   (naverage = 1)              Number of sample points to combine
   (niterate = 3)              Number of rejection iterations
 (low_reject = 3.0)            Low sigma rejection factor
(high_reject = 3.0)            High sigma rejection factor
       (grow = 0.0)            Rejection growing radius
       (mode = "ql")
```

9A) (optional) If you are having trouble getting CCDPROC to process and receive an error that there is a division by zero, you may have to run IMREPLACE to replace any negative values in your flat field. Here are the parameters of IMREPLACE that I used:

```
epar imreplace
       images = "normVflat"     Images to be edited
        value = 1.0             Replacement pixel value
   (imaginary = 0.0)            Imaginary component for complex
       (lower = INDEF)          Lower limit of replacement window
       (upper = 0.0)            Upper limit of replacement window
      (radius = 0.0)            Replacement radius
        (mode = "ql")
```

Do a quick IMSTAT on the corrected flat field to be sure that all the negative or 0 (zero) pixels have been eliminated (i.e. the MIN values are not negative or zero).

Now CCDPROC should complete its processing. Please check your processed images to make sure they look ok.

FRINGING IN RED FILTERS

Note that the I-band images (or any images taken in filters redder than the Johnson-Cousins R-band) have some fringing. Please check back once I've worked out how to deal with the fringing with ARCTIC data. There are some Fringe Flats taken by Dr. Russet McMillan from the summer of 2017 that you can use to try to remove the fringes yourself. Your mileage may vary since fringes vary throughout the night and are often caused by OH emission in the atmosphere. There is a nice description of fringing effects in Steve Howell's book, "Handbook of CCD Astronomy", pages 83-87, if you are unfamiliar with this effect.

FIGURES

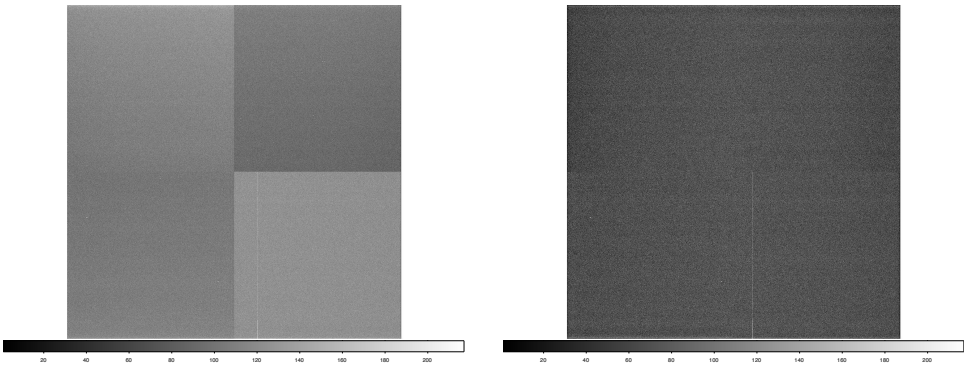

Figure 1: Left: Raw image of the bias in quad mode. Right: Trimmed and stitched bias. The overscan bias level has been removed.

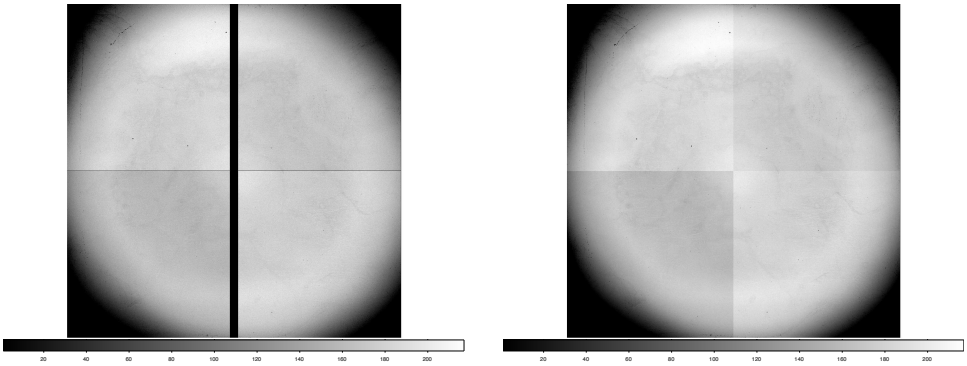

Figure 2: Left: Raw dome flat in V bandpass. Right: Trimmed and stitched V dome flat.

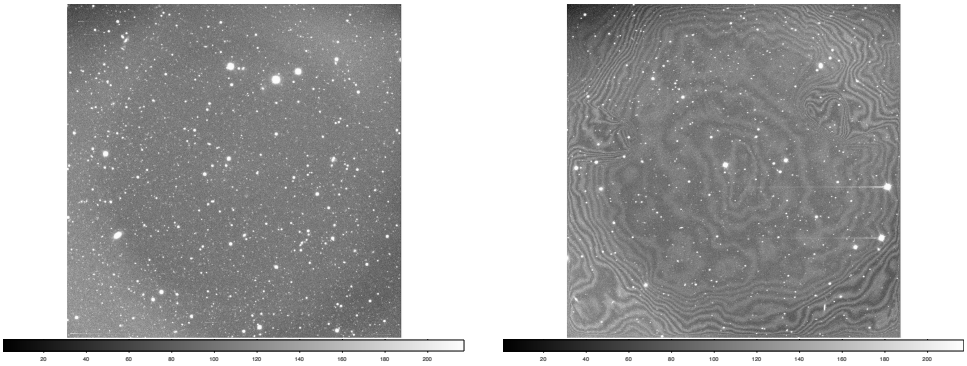

Figure 3: Left: Reduced V image. Right: Reduced I image. NB: Some illumination correction still needs to be applied due to the ring structure in the V image. This topic is still being investigated, so stay tuned.

Appendix

Here are some example scripts to separate, trim, and merge the quad-mode data from Steps 2-4. The first script is *dobias.cl*, which does these three tasks on the bias frames only. You can run each step separately or all in one go.

```
apobreak bias.0011.fits
apobreak bias.0012.fits
apobreak bias.0013.fits
apobreak bias.0014.fits
apobreak bias.0015.fits
apobreak bias.0016.fits
apobreak bias.0017.fits
apobreak bias.0018.fits
apobreak bias.0019.fits
apobreak bias.0020.fits

apotproc bias.0011.fits
apotproc bias.0012.fits
apotproc bias.0013.fits
apotproc bias.0014.fits
apotproc bias.0015.fits
apotproc bias.0016.fits
apotproc bias.0017.fits
apotproc bias.0018.fits
apotproc bias.0019.fits
apotproc bias.0020.fits

aponew bias.0011.fits
aponew bias.0012.fits
aponew bias.0013.fits
aponew bias.0014.fits
aponew bias.0015.fits
aponew bias.0016.fits
aponew bias.0017.fits
aponew bias.0018.fits
aponew bias.0019.fits
aponew bias.0020.fits
```

Here is the same script, but separate, trim, and merge tasks are done on my raw V sky flats. I called this script *dovflat.cl* and to run it at the IRAF prompt, type `cl < dovflat.cl`.

```
apobreak vflat.0042.fits
apobreak vflat.0043.fits
apobreak vflat.0044.fits
apobreak vflat.0045.fits
apobreak vflat.0046.fits

apotproc vflat.0042.fits
apotproc vflat.0043.fits
apotproc vflat.0044.fits
apotproc vflat.0045.fits
apotproc vflat.0046.fits

aponew vflat.0042.fits
aponew vflat.0043.fits
aponew vflat.0044.fits
aponew vflat.0045.fits
aponew vflat.0046.fits
```