Toward A More Perfect Flat-Field

We find that for our scientific goals it is necessary to improve the flat-field correction provided by our dome flats by generating a dark-sky or super-sky-flat. We generate this "delta" correction flat (which is really just accounting for color differences between the night-sky and our dome lamps, and for our case of trying to study objects at or below the sky level is appropriate to apply) by combining multiple object frames, effectively (ideally) rejecting all of the real objects in the frame and leaving us with a high SNR image of "blank" sky. Unfortunately, any pupil-ghost image or fringing (present in at least I-band, z'-band, and most red narrow-band filters) that might be present in your input images are still in this combined frame. Before you can apply a sky-flat to your images, and assuming you want to remove the additive pupil-ghost and fringing components from all of your images, you must first create template pupil-ghost and fringe-correction frames and subtract scaled versions of these frames from all of your object frames. If you can live with the errors introduced by just dividing these components out, you can just use what I call version one of the skyflat, produced by combining all of the images taken in a given filter over some period of time. For the NDWFS KPNO 4m data we potentially have to correct for both a pupil-ghost (depending on the band) and fringing (also wave-length dependent). For the CTIO data only fringing is a possible correction. For some bands, e.g. R-band, neither of these corrections is generally necessary. Below we take the most complicated/general case as an example: generating an I-band super-sky-flat. In order the steps that will be described are:

0.) Generating "object" masks for each image. These masks will be used to exclude from the combining process those regions of images that have significant light from resolvable/identifiable objects.

1.) Using 25 to 30 "object" frames along with **sflatcombine** and **mscpupil** to generate a Pupil template image that can be subtracted from each of the original object frames.

2.) How to subtract the pupil-ghost using **rmpupil** from each of your object frames.

3.) How to generate your fringe correction frame using 25-30 object frames from which the pupil-ghost has already been removed. You will use the tasks **sflatcombine**, **mscmedian**, **mscarith**, and **mscpupil**.

4.) How to remove the fringing from each of your object frames using the task **rmfringe**.

5.) How to generate your final super- or dark-sky-flat using **sflatcombine**.

Making the Pupil Template

The main "trick" we use in generating our dark-sky or "super-sky" flat is that we have many exposures in a given filter of many different regions of the sky. We combine these images, excluding objects, to get an image of the sky. In earlier versions of **mscred** we relied on having many images to allow us to "reject" objects. The latest version of IRAF and **mscred** enable a more sophisticated use of masks during the combining of images and new software for identifying "objects" and defects in images that we would want to exclude from a combined super-sky-flat. Before describing how we combine images to generate the pupil template, fringe template, and ultimately the super-sky-flat, we will describe how to generate the "object" masks for each image that will be used to help exclude objects from the combined sky-flat image.

In IRAF V2.11.3 the default mask type is pl files, which are created and placed into subdirectories. Masks in multi-extension fits files are not available. In our use of **ccdproc** earlier in this guide we generated masks of this pl type. Although in IRAF V2.12 the default file type for masks is multi-extension fits files, not all tasks, including **ccdproc** have been modified to create fits format mask files. Such tasks still generate the .pl mask type. Only the newest tasks in V2.12 generate .fits masks. To use pl files in subdirectories for both new and older tasks, at the cl prompt enter `set masktype = pl`. To switch back you can `set masktype = fits`.

Before you can generate the object masks, you will need to load the package **nproto**.

```
epar objmask

PACKAGE = nproto
   TASK = objmasks

images =        @obj.list  List of images or MEF files
objmasks =       @om.list  List of output object masks
(omtype =         numbers) Object mask type
(skys =         @sky.list) List of input/output sky maps
(sigmas =                ) List of input/output sigma maps
(masks =             !BPM) List of input bad pixel masks
(extnames =              ) Extension names
(logfiles =        STDOUT) List of log files\n
(blkstep =               1) Line step for sky sampling
(blksize =             -10) Sky block size (+=pixels, -=blocks)
(convolve =     block 3 3) Convolution kernel
(hsigma =                3.) Sigma threshold above sky
(lsigma =               10.) Sigma threshold below sky
(hdetect =              yes) Detect objects above sky?
```

```
(ldetect =             no) Detect objects below sky?
(neighbors =            8) Neighbor type"
(minpix =               6) Minimum number of pixels in detected objects
(ngrow =                2) Number of grow rings
(agrow =               2.) Area grow factor
(mode =                ql)
```

```
Where om.list:  and sky.list:

om092          sky092
om093          sky093
om094          sky094
om095          sky095
om096          sky096
om097          sky097
om098          sky098
etc.           etc.
```

You should edit the hidden parameter set **objmasks1** and change *fitstep = 10*, and *fitxorder* and *fityorder* = 1 before proceeding. `>epar objmasks1`

```
PACKAGE = nproto
   TASK = objmasks1

(exps   =                    ) List of exposure maps
(gains  =                    ) List of gain maps
(catalog=                    ) List of catalogs
(catdefs=                    ) List of catalog definitions
(dodetec=                  yes) Detect objects?
(dosplit=                   no) Split merged objects?
(dogrow =                  yes) Grow object regions?
(doevalu=                   no) Evaluate objects?
(skytype=                block) Type of sky estimation
(fitstep=                   10) Line step for sky sampling
(fitblk1=                   10) Block average for line fitting
(fithcli=                   2.) High sky clipping during 1D sky
estimation
(fitlcli=                   3.) Low sky clippling during 1D sky
estimation
(fitxord=                    1) Sky fitting x order
(fityord=                    1) Sky fitting y order
(fitxter=                 half) Sky fitting cross terms
(blknsub=                    2) Number of subblocks per axis
(updates=                  yes) Update sky during detection?
(sigavg =                   4.) Sigma of mean flux cutoff
(sigmax =                   4.) Sigma of maximum pixel
(bpval  =                INDEF) Output bad pixel value
(splitma=                INDEF) Maximum sigma above sky for splitting
(splitst=                  0.4) Splitting steps in convolved sigma
(splitth=                   5.) Splitting threshold in sigma
(sminpix=                    8) Minimum number of pixels in split objects
(ssigavg=                  10.) Sigma of mean flux cutoff
(ssigmax=                   5.) Sigma of maximum pixel
(magzero=                INDEF) Magnitude zero point
(mode   =                   ql)
```

For each image the task **objmask** combines the existing BPM masks (those generated by you earlier with **ccdproc**) with the new objects it finds in the image. We will expand the description of how this task works in the next version of this guide. Note that the object masks do not replace the BPM masks, as ultimately we will be combining subsets of these images to produce deep images in which we want to still have our objects! The "object" masks are just used for the combining steps described below.

To be able to subtract the pupil-ghost from the object frames we will need a template image that can be scaled and then subtracted from these frames. You will generate your pupil template using **mscpupil** and an input image that contains a high signal-to-noise ratio image of the pupil ghost. The "input image" can be a combined dome-flat from a narrow-band image close in wavelength to your observed band. We find, however, that the best results come from using an input image combined from many science frames of different (or dithered) regions of the sky, an initial "dark-sky" or "super-sky-flat" image. We will first describe how to generate such an input image, then describe how to use **mscpupil** to generate the template.

```
PACKAGE = mscred
   TASK = sflatcombine

input   =      @objects_I.list  List of images to combine
(output =         Sflat990327V1 Output sky flat field root name
(combine=              average) Type of combine operation
(reject =              ccdclip) Type of rejection
(ccdtype=               object) CCD image type to combine
(subsets=                  yes) Combine images by subset parameter?
(masktype=            !objmask) Mask type
(maskvalue=                 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)
```

The `objmask` keyword in the image headers should have been added by the **objmasks** task run earlier. Note that the **sflatcombine** task is likely to take several hours, even on a fast machine.

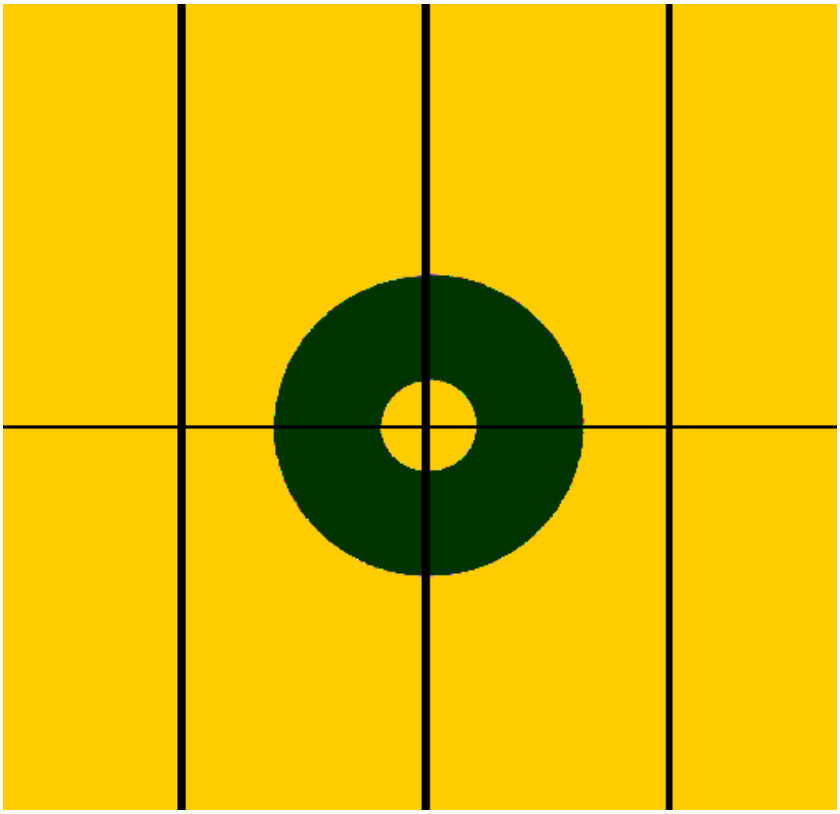
You used **mscpupil** when you corrected any dome-flat that had a pupil-ghost present. However, when you use the task to generate a pupil-template you need to save the measured "pupil-ghost". This requires setting `type=data` in the parameter file for mscpupil. Here is an example of suitable parameters for this step.

```
PACKAGE = mscred
TASK = mscpupil

input   =        Sflat990327V1I List of input images
output  =               Pupil_I List of output images
(masks  =                   BPM) List of masks
(type   =                  data) Output type
(lmedian=                    no) Subtract line-by-line median?
(xc     =                   27.) Pattern center offset (pixels)
(yc     =                    9.) Pattern center offset (pixels)
(rin    =                  300.) Radius of inner background ring (pixels)
(drin   =                   20.) Width of inner background ring (pixels)
(rout   =                 1500.) Radius of outer background ring (pixels)
(drout  =                   20.) Width of outer background ring (pixels)
(funcin =             chebyshev) Inner azimuthal background fitting
function
(orderin=                    2) Inner azimuthal background fitting order
(funcout=               spline3) Outer azimuthal background fitting
function
(orderou=                    2) Outer azimuthal background fitting order
(rfuncti=               spline3) Radial profile fitting function
(rorder =                   40) Radial profile fitting order
(abin   =                   0.) Azimuthal bin (deg)
(astep  =                   0.) Azimuthal step (deg)
(niterat=                    3) Number of rejection iterations
(lreject=                   3.) Low rejection rms factor
(hreject=                   3.) High rejection rms factor
(datamin=                INDEF) Minimum good data value
(datamax=                INDEF) Maximum good data value
(verbose=                  yes) Print information?
(fd     =                    )
(mode   =                   ql)
```

Note that the input image for **mscpupil** needs to have a high SNR image of the pupil-ghost and not have objects or other light in the frame (if the frames have fringing it will not be possible to avoid that contribution). For the NDWFS, we generate this image, in the above example named Sflat990327V1I (indicating the first version, V1, of a sky-flat for I-band data taken on March 27, 1999 UT), by combining 25 to 35 individual object frames that have been processed through the application of the dome-flat. This produces an image with high SNR in the sky, pupil-ghost, and fringing. The output file, Pupil_I in the example above, will be an

image containing the pupil-ghost, some fringing signal in the region of the pupil-ghost, plus some defects due to bad columns in the region of the pupil-ghost. This image will be scaled and subtracted from the individual object frames using **rmpupil**. Here is an example of the parameters for **sflatcombine** to generate the input image for **mscpupil**. The asci file input list of images to **sflatcombine** contains one image per line. We usually use 25 to 30 images. We have successfully used imags from two similar nights if on a single night we did not have enough images. Note that **sflatcombine** will append the filter name to the end of the root name you specify, so the output of the task below would be Sflat990327V1I.fits.

Now you need to create a mask for the pupil template so that only the relevant parts of it are used later when we use it to remove the pupil ghost from our science frames. This is fast step. We use **mscpupil** again, but with `type= mask`. This only has to be done once for all runs and filters with a given instrument. This step is very quick and you can run it easily using the parameters below for your data. We will eventually add this mask to the calibration data web page. For now, here is an example for Mosaic-1 used at the Kitt Peak 4m:

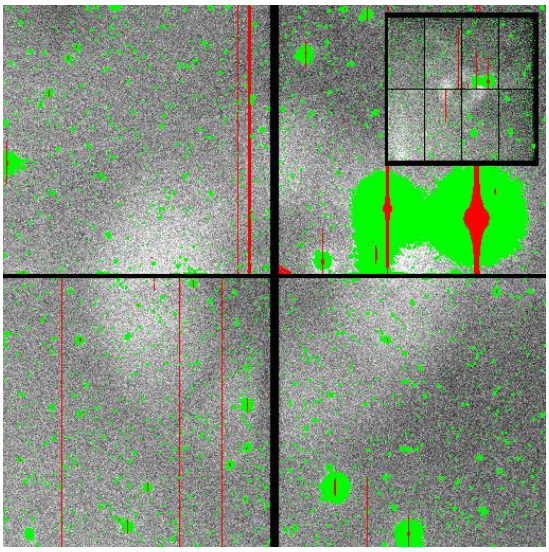


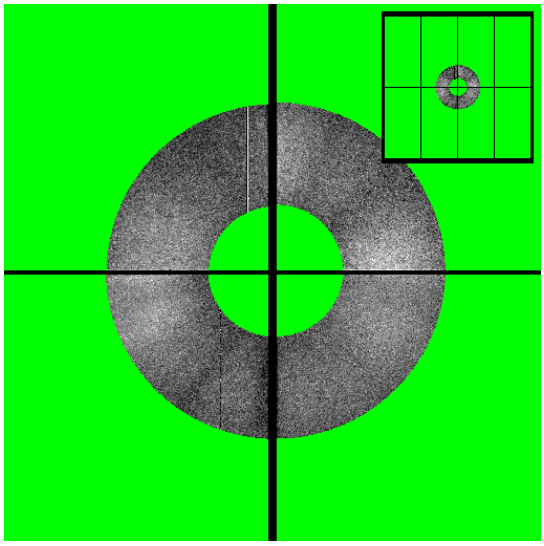
To download this image press the shift key as you click on this link: Pupil_mask.fits

```
PACKAGE = mscred
   TASK = mscpupil

input = "Pupil_I.fits"     List of input images
output = "Pupil_mask"      List of output images
(masks = "")               List of masks
(type = "mask")            Output type
(lmedian = no)             Subtract line-by-line median?
(xc = 27.)                 Pattern center offset (pixels)
(yc = 9.)                  Pattern center offset (pixels)
(rin = 405.)               Radius of inner background ring (pixels)
(drin = 20.)               Width of inner background ring (pixels)
(rout = 1350.)             Radius of outer background ring (pixels)
(drout = 20.)              Width of outer background ring (pixels)
(funcin = "chebyshev")     Inner azimuthal background fitting function
(orderin = 2)              Inner azimuthal background fitting order
(funcout = "spline3")      Outer azimuthal background fitting function
(orderout = 2)             Outer azimuthal background fitting order
(rfunction = "spline3")    Radial profile fitting function
(rorder = 40)              Radial profile fitting order
(abin = 0.)                Azimuthal bin (deg)
(astep = 0.)               Azimuthal step (deg)
(niterate = 3)             Number of rejection iterations
(lreject = 3.)             Low rejection rms factor
(hreject = 3.)             High rejection rms factor
(datamin = INDEF)          Minimum good data value
(datamax = INDEF)          Maximum good data value
(verbose = yes)            Print information?
(fd = "")
(mode = "ql")
```

Here are some example images created by Frank Valdes:

Object and pupil template with associated masks





Removing the Pupil-Ghost

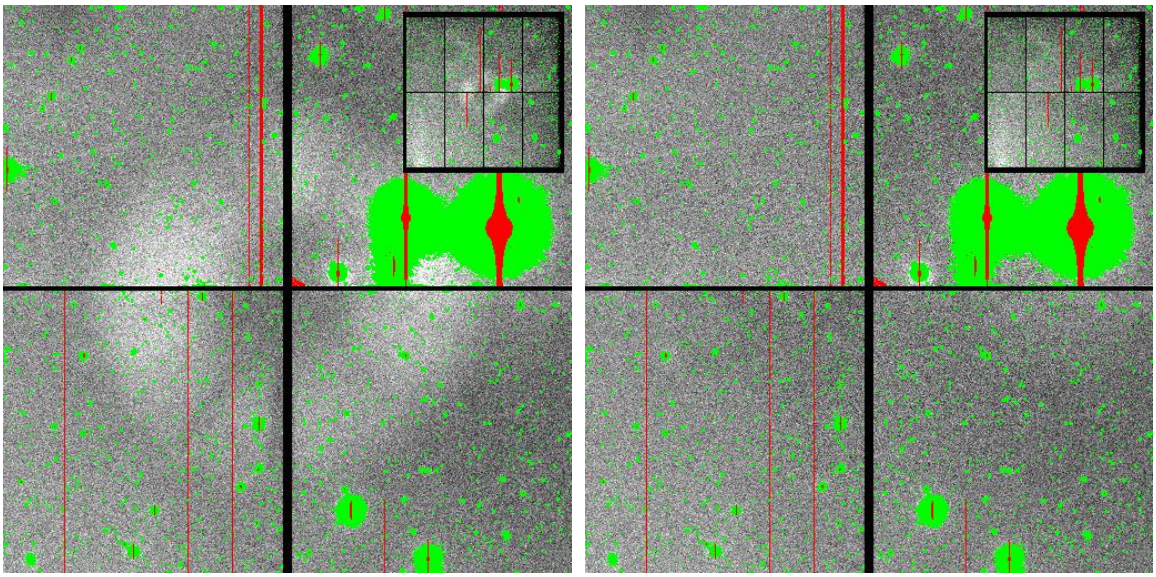
Now **rmpupil** can be used to remove the pupil-ghost from the individual object frames. This routine scales the pupil template generated above, (`Pupil_I.fits`), and subtracts it out of the input image. Prior to the current version (4.7) of **mscred**, we found it necessary to adjust the scaling interactively (the interactive version of **rmpupil** is now called **irmpupil** in version 4.7 of **mscred**. In previous versions, **rmpupil** was interactive). We now find the non-interactive mode to be very robust on our data. However, we encourage you to inspect the results to verify that the task is working well on your data as we have not been able to test this new version on a diverse set of images. For a description of how the interactive mode worked with the old version of this task, see version 6.2 of this guide. Below is a sample parameter listing for the new **rmpupil**. Note that we have chosen to create new output images and to indicate that they have been corrected for the pupil-ghost by adding a "p" to its name. The name of the file used as the template as well as the adopted scaling are written to the image header.

```
PACKAGE = mscred
TASK = rmpupil

input   =            @obj.list  List of input mosaic exposures
output  =           @objp.list  List of output mosaic exposures
pupil   =              Pupil_I  Pupil orlist of pupil patterns
masks   =            @om.list  List of object/bad data masks
(pupilmasks =       Pupil_mask) Pupil masks
(outtype=                sdiff) Output type
(ncblk  =                    3) Column smoothing for weights
(nlblk  =                    3) Line smoothing for weights
(extfit =           "im[2367]") Extensions to use in scaling fit
(logfile=                   "") Logfile
(verbose=                  yes) Verbose?
(mode   =                 "ql")
```

Inspect the corrected images to verify that the correction was done correctly, i.e. that the pupil ghost has been removed.

Example images of the pupil ghost have been created by Frank Valdes. Note in these examples the object masks have been overlaid.

Before and after pupil removal



Making The Fringe Correction Frame

Just as we had to remove the pupil-ghost, we might have to remove the contribution of fringing from our frames. In a manner similar to that followed to correct for the pupil-ghost we combine multiple object frames (this time those that have had the pupil-ghost removed) to produce a new image from which we will construct a fringe template. This is also a good time to check the accuracy of our subtraction of the pupil-ghost, since if all the images have been handled properly there will be no pupil in this new combined sky-flat. Note that the removal of the pupil-ghost will have partially (in some cases completely) corrected the fringing in the region of the pupil-ghost. Here is a parameter file for using **sflatcombine** to generate the second-pass version (no pupil-ghost, fringing still present) of the sky-flat. The output image in this example will be named Sflat990327V2I.fits.

```
PACKAGE = mscred
   TASK = sflatcombine

input   =   @object_p_I.inlist  List of images to combine
(output =        Sflat990327V2) Output sky flat field root name
(combine=              average) Type of combine operation
(reject =              ccdclip) Type of rejection
(ccdtype=               object) CCD image type to combine
(subsets=                  yes) Combine images by subset parameter?
(masktype=            !objmask) Mask type
(maskvalue=                 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)
```

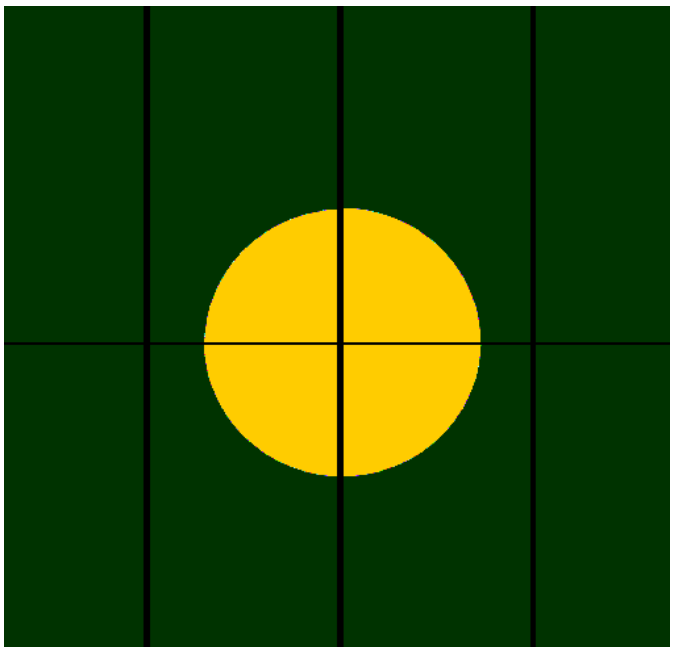
There are several remaining steps required to finally generate the fringe correction frame. We want our fringe template to only have the signal from the fringing, not the residual flat-field structure we want to correct with our sky-flat. For this reason, we run **mscmedian** on the combined image to generate an image with only the large-spatial-scale features we do not want in the fringe template, and subtract this from our initial V2 "sky-flat" (fringe frame) to generate out final fringe template image. An example parameter file for **mscmedian** is below.

```
PACKAGE = mscred
   TASK = mscmedian

input = "Sflat990327V2I"  Input mosaic images
output = "MedianTempI"    Output mosaic images
xwindow = 129             X window size of median filter
ywindow = 129             Y window size of median filter
(outtype = "median")      Output type (median|difference)
(zloreject = -20000.)     Lowside pixel value cutoff
(zhireject = 30000.)      High side pixel value cutoff
(verbose = yes)           Print messages about actions taken by the task
(fmedian = yes)           Use fast median algorithm?
(hmin = -20000)           Minimum histogram bin
(hmax = 30000)            Maximum histogram bin
(zmin = -20000.)          Pixel value corresponding to hmin
(zmax = 30000.)           Pixel value corresponding to hmax
(fd = "")
(mode = "ql")
```

This task takes a significant amount of time -- around 90-120 minutes on a Sun ultra-60. The generation of the finge template can now be completed using **mscarith**.

```
PACKAGE = mscred
   TASK = mscarith

operand1=  Sflat990327V2I.fits  Operand image or numerical constant
op      =                    -  Operator
operand2=     MedianTempI.fits  Operand image or numerical constant
result  =     FringeI.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)
```

As we did for the pupil template, we now need to make a mask inidicating the regions to be used later for the subtraction. For this mask, we want to use only those extensions not affected by the pupil pattern for estimating the the fringe scaling. This is discussed in more detail here. We will eventually add this fringe mask to the calibration data web page. For now, here is an example for Mosaic-1 used at the Kitt Peak 4m:



To download this image press the shift key as you click on this link: Fringe_mask.fits

```
PACKAGE = mscred
   TASK = mscpupil

input   =        FringeI.fits  List of input images
output  =          Fringe_mask  List of output images
(masks  =                     ) List of masks
(type   =                 mask) Output type
(lmedian=                   no) Subtract line-by-line median?
(xc     =                  27.) Pattern center offset (pixels)
(yc     =                   9.) Pattern center offset (pixels)
(rin    =                1500.) Radius of inner background ring (pixels)
(drin   =                  20.) Width of inner background ring (pixels)
(rout   =                8000.) Radius of outer background ring (pixels)
(drout  =                  20.) Width of outer background ring (pixels)
(funcin =           chebyshev) Inner azimuthal background fitting
function
(orderin=                   2) Inner azimuthal background fitting order
(funcout=             spline3) Outer azimuthal background fitting
function
(orderou=                   2) Outer azimuthal background fitting order
(rfuncti=             spline3) Radial profile fitting function
(rorder =                  40) Radial profile fitting order
(abin   =                  0.) Azimuthal bin (deg)
(astep  =                  0.) Azimuthal step (deg)
(niterat=                   3) Number of rejection iterations
(lreject=                  3.) Low rejection rms factor
(hreject=                  3.) High rejection rms factor
(datamin=                INDEF) Minimum good data value
(datamax=                INDEF) Maximum good data value
(verbose=                  yes) Print information?
(fd     =                    )
(mode   =                   ql)
```

Subtracting the Fringe Template from Your Frames

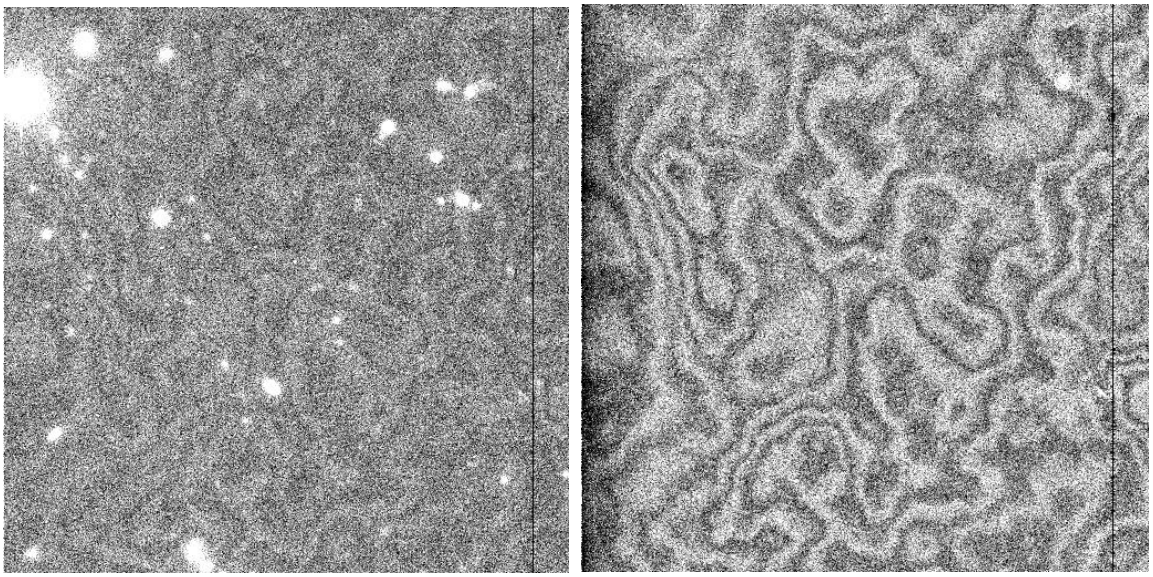
The task **rmfringe** can now be run on each object frame. This task works in a similar manner to **rmpupil**, but works on all 8 CCDs rather than just the central four. Like **rmpupil**, we used to run the old version of this task interactively, but the improved automated version of this task seems to work very well on images like those taken for the NDWFS (i.e. fields without large spatially extended sources) and we recommend its use for similar images (the interactive version of **rmfringe** is called **irmfringe** in version 4.7 of **mscred**. In previous versions, **rmfringe** could be interactive). I (BTJ and Valdes) would welcome hearing about success using and/or problems with the new automated version of the task.

```
PACKAGE = mscred
   TASK = rmfringe

input   =           @objp.list  List of input images
output  =          @objpf.list  List of output corrected images
fringe  =         FringeI.fits  Fringe or list of fringe patterns
masks   =             @om.list  List of object/bad data masks
(fringem=          Fringe_mask) Fringe masks
(backgro=            @sky.list) Lisk of input image backgrounds
(extfit =                     ) Extensions to use in scaling fit
(logfile=                     ) Logfile
(verbose=                  yes) Verbose?
(mode   =                   ql)
```

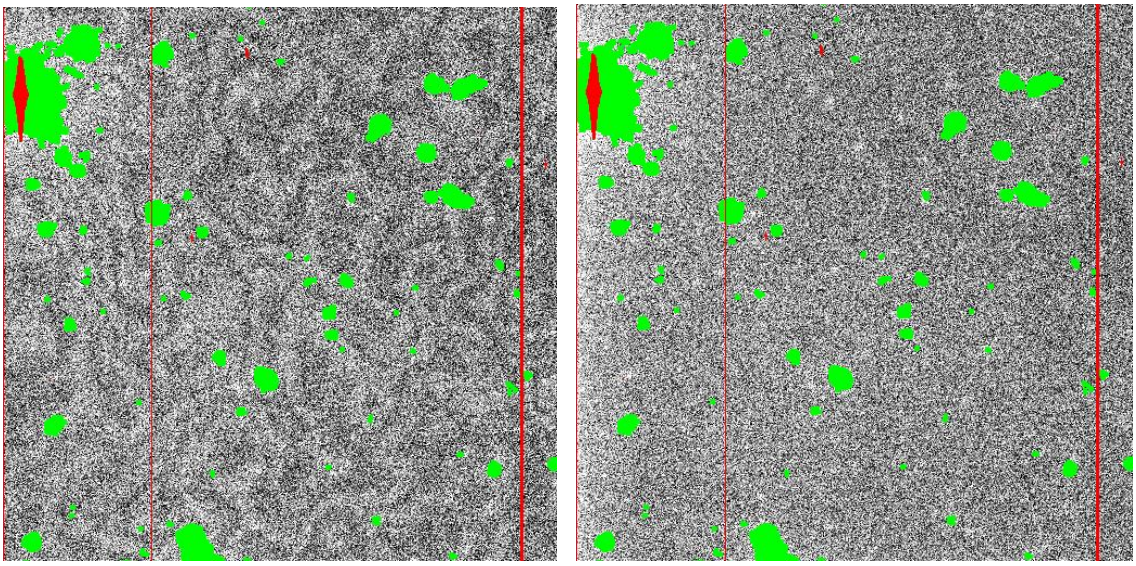
Note that the output file is a new image, in this case we choose to append an "f" to the existing file name.

Below are some example images created by Frank Valdes. Note in the "Before" and "After" frames the object masks have been overlaid.

Example of fringing



Before and after comparison of fringe removal.



Generating Your Final Sky-Flat

Now that you have versions of all your object frames that are free of pupil-ghost images and the effects of fringing, you can finally run **sflatcombine** one last time to generate your super- or dark-sky-flat. In the case of I-band data, this is the third time you will have run **sflatcombine**. Things would have been easier if you were working in the R-band. In that case you have no significant pupil-ghost to correct and no fringing, so the product of your first **sflatcombine** is likely to have been suitable to use as your sky-flat. Here is the parameter file for your third pass of **sflatcombine**:

```
PACKAGE = mscred
TASK = sflatcombine

input   =    @object_pf_I.list  List of images to combine
(output =        Sflat990327V3) Output sky flat field root name
(combine=              average) Type of combine operation
(reject =              ccdclip) Type of rejection
(ccdtype=               object) CCD image type to combine
(subsets=                  yes) Combine images by subset parameter?
(masktype=            !objmask) Mask type
(maskvalue=                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)
```

Now it is finally time to apply the sky flat (Sflat990327V3I.fits in this example) to the object frames that have had the pupil-ghost and fringe components subtracted (our obj*pf.fits files). This will be done using **ccdproc** (see next section of this guide).