Fringing correction: creating masters and applying to data

Information originates from (and there is lots of more information!): http://www.ifa.hawaii.edu/~rgal/science/lfcred/lfc_red.html#fringe

Prerequisites

- IRAF installed
- MSCRED package installed. MSCRED translates to MoSaiCREDuction.MSCRED package can be installed easily nowadays using dynamic extern package loading mechanism built in to IRAF since v2.15, if not familiar with that, see: /iraf/iraf.v2161/extern/README (or whatever the IRAF installation directory is).
- Blank sky images (with as few as possible non-saturated stars on them) OR scientific targets from several/many sky areas (so that objects on them do not overlap).
- Correctly preprocessed data, using good quality calibration files - bias, dark, and flat frames.
- All blank sky images should be properly flatfielded, too.
- Properly set up FITS headers. Most important for that analysis are EXPTIME, FILTERS, and IMAGETYP, but also GAIN and RDNOISE values.
- Processing will be done completely independently in all of the problematic filters. I <> R <> Halpha <> i <> z.
- **VERY IMPORTANT: object mask file type must be pl.**

```
set masktype = pl
```

Creating fringe master file(s)

First step would be preprocessing of the data - using bias, dark, flat and what else might be needed. I have used **ccdred** package in IRAF to create all calibration master files and to apply needed calibrations
NB!!! Blank sky files shall be flatfielded, too!

It seems (to me) that it is easiest to proceed in IRAF, using lists. Filenames of object mask and sky files could start with om and sky.. Because sed should be installed in every Unix computer, it's convenient to make those changes like this:

```
ls myblankskyfile*.fits > targets.list
!sed 's/\.fits//g' targets.list | sed 's/^/om/g' > om.lst
!sed 's/^om/sky/g' om.lst > sky.lst
```

And now let's prepare lists for scientific target(s) in the same way we did for blank sky frames:

```
ls SN2016bkv-BIN2-00*-I*.fits > sn.list
!sed 's/\.fits//g' sn.list | sed 's/^/om/g' > omsn.list
!sed 's/^om/sky/g' omsn.list > skysn.list
```

Now entering to mscred package:

```
mscred
```

.. and starting to set things up, first "superflatcombine". sflatcombine is combining images of e.g. **blank sky areas** OR **many scientific target frames** that have enough signal in the background and fringing patterns. In the second case, it works only if there is enough information for every pixel. If one area is well guided, all the stars are on the same location on the sensor and resulting combined sky flat has holes at those positions. sflatcombine uses object masks to exclude pixels with stars and/or other objects.

```
epar sflatcombine
```

```
 PACKAGE = mscred
    TASK = sflatcombine
input   =        @targets.list  List of images to combine
(output =                Sflat) Output sky flat field root name
(combine=              average) Type of combine operation
(reject =            avsigclip) Type of rejection
(ccdtype=                    ) CCD image type to combine
(subsets=                  yes) Combine images by subset parameter?
(masktyp=             !objmask) Mask type
(maskval=                   0.) Mask value
(scale  =                 mode) Image scaling
(statsec=                    ) Image section for computing statistics
(nkeep  =                    1) Minimum to keep (pos) or maximum to reject
(neg)
(nlow   =                    1) minmax: Number of low pixels to reject
(nhigh  =                    1) minmax: Number of high pixels to reject
(mclip  =                  yes) Use median in sigma clipping algorithms?
(lsigma =                   6.) Lower sigma clipping factor
(hsigma =                   3.) Upper sigma clipping factor
(rdnoise=              rdnoise) ccdclip: CCD readout noise (electrons)
(gain   =                 gain) ccdclip: CCD gain (electrons/DN)
(snoise =                   0.) ccdclip: Sensitivity noise (fraction)
(pclip  =                 -0.5) pclip: Percentile clipping parameter
(blank  =                   1.) Value if there are no pixels
(grow   =                   3.) Radius (pixels) for neighbor rejection
(fd     =                     )
(mode   =                   ql)
```

When fringing correction master file is being done, mean signal level in the background must be subtracted. Therefore, **median filtering of combined sflat files IS PARTICULARILY IMPORTANT!!** Final fringing correction depends on that this step quite strongly, so e.g. **[xy]window** s hould be changed as appropriate, according your data and filtered result.

```
epar mscmedian
```

```
PACKAGE = mscred
   TASK = mscmedian
input   =               SflatI  Input mosaic images
output  =            MedSFlatI  Output mosaic images
xwindow =                  328  X window size of median filter
ywindow =                  328  Y window size of median filter
(outtype=               median) Output type (median|difference)
(zloreje=             -20000.) Lowside pixel value cutoff
(zhireje=              30000.) High side pixel value cutoff
(verbose=                 yes) Print messages about actions taken by the
task
                               # Fast median
(fmedian=                 yes) Use fast median algorithm?
(hmin   =              -32768) Minimum histogram bin
(hmax   =               32767) Maximum histogram bin
(zmin   =             -20000.) Pixel value corresponding to hmin
(zmax   =              30000.) Pixel value corresponding to hmax
(fd     =                   )
(mode   =                  ql)
```

In our simple case (i.e. no actual mosaic camera) subtraction of filtered sflat from sflat can be done manually (just using imarith), but just for convenience, we can use set up and use also:

```
epar mscarith
```

```
PACKAGE = mscred
   TASK = mscarith
operand1=               SflatI  Operand image or numerical constant
op      =                    -  Operator
operand2=            MedSFlatI  Operand image or numerical constant
result  =              FringeI  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=                 yes) Print operations?
(noact  =                  no) Print operations without performing them?
(fd1    =                   )
(fd2    =                   )
(fd3    =                   )
(mode   =                  ql)
```

VERY IMPORTANT step: Fringing correction depends absolutely on creation of good objects masks for both blank sky frames and later object frames. Fine tune the parameters of tasks **objmasks1** and **objmasks** to get satisfying results. Defaults are close, but not necessarily good enough.

```
nproto
set masktype=pl
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=                     2) Sky fitting x order
(fityord=                     2) 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)
```

There are wheels inside the wheels, a bit similarly to IRAF daophot package, typically one should change at least **hsigma**:

```
epar objmasks
```

```
PACKAGE = nproto
   TASK = objmasks
images  =        @targets.list  List of images or MEF files
objmasks=              @om.lst  List of output object masks
(omtype =              numbers) Object mask type
(skys   =             @sky.lst) List of input/output sky maps
(sigmas =                    ) List of input/output sigma maps
(masks  =                !BPM) List of input bad pixel masks
(extname=                    ) Extension names
(logfile=               STDOUT) List of log files
(blkstep=                    1) Line step for sky sampling
(blksize=                  -10) Sky block size (+=pixels, -=blocks)
(convolv=            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?
(neighbo=                    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)
```

Now, creating object mask and sky files is trivial. It is very good to look at the output, one should not see any errors.Turning on logging to file (setting parameter logfile) may be useful.

```
objmasks
või
objmasks @targets.list objmasks=@om.lst skys=@sky.lst hsigma=4.0
```

Created mask files can be displayed as ordinary files in DS9. In general, it is a good idea to load object mask file and corresponding blank sky frame to ensure that background and fringing are not included in masks and as many stars as possible are included. Loading them to different DS9 channels and blinking between them works well. E.g.:

```
display myblankskyfile001 1
display ommyblankskyfile001.pl 2
või
display myblankskyfile overlay=ommyblankskyfile.pl bpm=BPM bpdisp=overlay bpcolor=red
frame=1
```

Now blank sky files can be combined to "superflat" (or "skyflat" :-) ?)
NB!!! this procedure can also be used to create illumination correction for twilight or panel/screen flats.

Task sflatcombine honours filters, if subsets are enabled (default they are) and correct filter keywords are set up in FITS files. See e.g. output using **ccdlist**. [I] shows that image has taken through I filter.

```
mscred> ccdlist blank-BIN2-001-I-120s-wcs.fits
blank-BIN2-001-I-120s-wcs.fits[1024,1024][real][][][I][OTZF]:CAblank2
```

```
mscred
epar sflatcombine
sflatcombine
```

Check if created sflat looks ok, without any stars.

```
display SflatI.fits
implot SflatI.fits
```

Now, lets smoothing Sflat* and checking the result (and adjusting parameters of mscmedian, when needed):

```
mscmedian
display MedSFlatI.fits
```

Subtracting median file from Sflat - because fringing is ADDITIVE effect - can be done using mscarith (or in non-mosaic case, imarith). The result of this step is our **fringing correction file.**

```
mscarith
display FringeI.fits
```

Applying fringe correction to science data

Now, to remove fringing from science frames, task **rmfringe** is used. Also as in the fringe frame creation process, masks are very important. It is extremely desirable to exclude stars and other interesting objects from fringe master frame scaling process. Otherwise stars, galaxies etc may affect scaling and cause unpredictable results. It may be useful to turn logging on (set logfile in epar rmfringe or from command line) - for documenting purposes.

```
epar rmfringe
```

```
PACKAGE = mscred
   TASK = rmfringe


input   =        @targets.list  List of input images
output  =   fr_//@targets.list  List of output corrected images
fringe  =              FringeI  Fringe or list of fringe patterns
masks   =            @om.list  List of object/bad data masks
(fringem=                    ) Fringe masks
(backgro=           @sky.list) Lisk of input image backgrounds
(ncblk  =                   5) Column smoothing
(nlblk  =                   5) Line smoothing
(extfit =                    ) Extensions to use in scaling fit
(logfile=                    ) Logfile
(verbose=                 yes) Verbose?
(mode   =                  ql)
```

Now, it is last moment to prepare target lists (objects, masks, skys), it they already havent been done.

And, when executing **rmfringe:**

```
rmfringe
```

We should get result(s) as fringe-corrected frame(s).

Faster way for that (at least easier to just copy-paste when everything is set up) is to use parameters on command line. Pay attention to **hsigma** e tc parameters that may affect creation of object masks!

```
objmasks @sn.list objmasks=@omsn.list skys=@skysn.list hsigma=4.0
rmfringe @sn.list output=fr_@sn.list masks=@omsn.list backgro=@skysn.list
```

Final note: **help rmfringe** is actually a very good source of information - with good examples.