SDFRED
Data Reduction Software for Subaru Suprime-Cam

Manual Version 1.4

Masami Ouchi
National Astronomical Observatory of Japan/ Space Telescope Science Institute

August 6, 2004 created

Translation by Catherine Ishida
National Astronomical Observatory of Japan

December 12, 2006 revised (Hisanori Furusawa, Yutaka Komiyama, YAGI,Masafumi & SDFRED support team)

September 25, 2007 revised (Hisanori Furusawa, YAGI,Masafumi & SDFRED support team)

May 1, 2008 revised (YAGI,Masafumi & SDFRED support team)

September 1, 2008 revised (YAGI,Masafumi & SDFRED support team)

This software manual is valid for reducing data observed before 2008/07/21

Introduction

Suprime-Cam is a mosaic CCD camera with 10 2048x4096 pixels CCDs. SDFRED, the Suprime-Cam Deep field REDuction package, is a software package for reducing Suprime-Cam data. SDFRED allows the immense data output from Suprime-Cam to be reduced semi-automatically, or even fully automatically, using typical input parameters. The reduced images are ready for scientific analysis.

SDFRED aimed at reducing deep field or blank field images. Images that contain objects that occupy a significant portion of a chip and/or too crowded regions may not be reduced properly with this package. The most problematic procedures are noted in the text.

Copyright

Ouchi et al., 2004, ApJ, 611, 660

How to Obtain SDFRED

The software package is available at:

http://optik2.mtk.nao.ac.jp/subaru_red/SPCAM/

Updates and other information concerning the package will be posted to this web page.

Other Software Requirements

SExtractor
http://terapix.iap.fr/rubrique.php?id_rubrique=91
(Bertin & Arnouts 1996, A&AS, 117 393)

The latest SDFRED (20080610) has been tested with version 2.5.0

IRAF(Not necessary for SDFRED, but useful)
http://iraf.nao.ac.jp/ or_ http://iraf.net/

The latest SDFRED (20080610) has been tested with version 2.12.2

Other basic software packages:
C compiler, perl, csh, sh/bash

Computer Hardware and Operating System Requirements

SDFRED requires a UNIX based computer with at least 256MB of memory. The necessary hard drive space depends on the quantity of data, but several to several hundred GB is generally required. (To reduce the practice dataset, 10GB is sufficient.)

The latest software (20080610) has been tested with Linux.

Installation

1. Uncompress the software package in an appropriate directory.

Download the latest version at the download URL written above.

```
$ tar xvzf sdfred20080610.tar.gz

or

$ gunzip -c sdfred20080610.tar.gz | tar xvf -
```

(In this text, % or $ at the beginning of a line represents a shell prompt.)

A directory with a name like sdfred20080610/ will be created in the current directory. The number represents the software version.

2. Compile

Move (cd) into the the sdfred20080610 directory and build the software.

```
$ cd sdfred20080610/
$ ./configure
$ make all
```

Then programs are installed into sdfred20080610/bin/

3. Add directory to PATH

Use an editor to add the sdfred20080610/bin directory (executables directory) to your PATH environment in your shell configuration file. In this example, /wa03a/subaru20/sdfred20080610/bin is the directory where SDFRED binaries are installed. You should modify as your environment.

bash users:

Edit ~/.bashrc *and* ~/.bash_profile as follows;

```
Example:

$ emacs ~/.bashrc
$ emacs ~/.bash_profile
```

```
PATH=/wa03a/subaru20/sdfred20080610/bin:$PATH
export PATH
```

~/.bashrc is used when new terminal is open, and ~/.bash_profile is used when you log into the computer. Therefore you need to modify both files.

After the files are updated, reflect the new setting to the current environment. This procedure is required only when you change the shell configuration files.

```
$ source ~/.bashrc
```

csh/tcsh users:

Edit ~/.cshrc to add the directory where SDFRED binaries are installed (executables directory) as follows;

```
Example:

% emacs ~/.cshrc
```

```
set path=( /wa03a/subaru20/sdfred20080610/bin $path )
```

After the files are updated, reflect the new setting to the current environment. This procedure is required only when you change the shell configuration files.

```
% source ~/.cshrc
% rehash
```

4. Set environmental variables

LANG, and LC_ALL should be set as "C", so that shell scripts and perl scripts work correctly

bash users:

Again, edit ~/.bashrc *and* ~/.bash_profile as follows;

```
$ emacs ~/.bashrc
$ emacs ~/.bash_profile
```

Add following two lines.

```
export LANG=C
export LC_ALL=C
```

And reflect the setting to the current session.

```
$ source ~/.bashrc
```

csh/tcsh users:

Again, edit ~/.cshrc as follows;

```
Example:

% emacs ~/.cshrc
```

Add following two lines.

```
setenv LANG C
setenv LC_ALL C
```

And reflect the setting to the current session.

```
% source ~/.cshrc
```

5. Other settings

If you use IRAF, you need to execute mkiraf in your "work directory".

Other additional setting may be required depends on your computer environment

When you have finished the setting, check the environment as follows

```
Example:

$ env
$ which namechange.csh
```

In env command result, check LANG, and LC_ALL environment.

And "which" command found the directory you have installed, the setting is OK

Once you finish the software preparation and setting, you can start from the next section next time.

Preparation of Data

Before staring a reduction, you need to sort data files into directories.

SDFRED requires that all input files must be in the "current directory." If you want to process some files in a different directory, the files should be recognized as if they are in the "current directory", by making symbolic links, or by other way.

Practice Dataset

A practice dataset (spcam_training_data.tar.gz; 753MB uncompressed) is available at

http://optik2.mtk.nao.ac.jp/subaru_red/SPCAM/
(spcam_training_data.tar.gz)

We have checked that the software works well for the dataset.

The specific examples given in the manual refer to this practice dataset.

```
Example:
extract practice data
$ tar xvzf spcam_training_data.tar.gz
```

After the data extraction, check the images; Which are the object frames? Which can be used as flat? Which are the standard star frames?

Also, examine images by eye-inspection using saoimage-ds9 or other your favorite FITS browser. Checkpoint is, for example, "are there any files with distorted/elongated stars?" If you have observation night log, refer to it.

SUPA0020*.fits are the object frames (target object), and SUPA0019512*.fits are the standard frames in the practice dataset.

In this version (ver1.4), the data are assumed to be reduced/analyzed in two directories (object/ and standard/) which are in the same directory as spcam_training_data.

```
---(work directory root) - spcam_training_data/
                         - object/
                         - standard/
```

An example of making symbolic links:

```
$ mkdir object standard
```

The result of ls is like this;

```
$ ls
object spcam_training_data spcam_training_data.tar.gz standard
```

Then link object frames into object/ directory

```
$ cd object/
$ ln -s ../spcam_training_data/SUPA0020*.fits .
```

Copy blank map data

```
$ cp ../spcam_training_data/blankmap* .
```

Link standard object frames into standard/ directory

```
$ cd ../standard/
$ ln -s ../spcam_training_data/SUPA0019512?.fits .
```

Check that 10 FITS files are in standard/ directory, and 50 FITS files and 6 blankmaps in object/ directory.

Retrieval from smoka

All data obtained at the Subaru are available at smoka public archive after 1.5 year of the proprietary period.

http://smoka.nao.ac.jp/

You can reduce the data from smoka with SDFRED.

In this manual, the target object and the standard object are reduced in different directories.

Check that data in a directory have observed with the same filter (e.g. V-band), and be the same data type (object/standard). Otherwise, you need to separate them into directories.

Data Reduction Sequence

Target Object frames:

Step		Command	time	Files Generated
(1)	Renaming	namechange.csh	20s,1m	(H*.fits)
(2)	Overscan and bias subtraction	overscansub.csh	1s,1s	(To_TH*.fits)
(3)	Making flat frames	mask_mkflat_HA.csh	6m,20m	(Flat frames: *mflats*.fits)
(4)	Flat fielding	ffield.csh	30s,2m	(fTo_RH*.fits)

(5)	Distortion correction and			
	atmospheric dispersion correction	distcorr.csh	1m,6m	(gfTo_RH*.fits)
(6)	PSF size measurement	fwhmpsf.csh	*,*	---
(7)	PSF size equalization	psfmatch_batch.csh	40m,80m	(pgfTo_RH*.fits)
(8)	Sky subtraction	skysb.csh	1m,3m	(spgfTo_RH*.fits)
(9)	Masking out AG probe	mask_AGX.csh	15s,2m	(AspgfTo_RH*.fits)
(10)	Masking out bad regions	blank.csh, ...	15s,1m	(bAspgfTo_RH*.fits)
(11)	Alignment	makemos.csh	2m,15m	(Alignment file: *.mos)
(12)	Coadding	imcio2a	3m,12m	(Final Image)

Standard Object frames:

Step	Command	Files Generated	
(1S)	Renaming	namechange.csh	(H*.fits)
(2S)	Overscan and bias subtraction	overscansub.csh	(To_TH*.fits)
(3S)	Flat fielding	ffield.csh	(fTo_RH*.fits)
(4S)	Distortion correction and		
	atmospheric dispersion correction	distcorr.csh	(gfTo_RH*.fits)
(5S)	Relative gain correction	--	

Each of the processes will apply to a different subset of data frames. The basic data flow consists of making lists of these various subsets, and providing these lists to various programs within the SDFRED package. Each step produces a new set of images or data files. The naming convention for the various new files are in parenthesis after each step in the list above.

The times listed in the table are how long it takes to reduce the practice dataset(5 exposures), and a bit larger set(17 exposures), respectively. The time is measured using a Intel Xeon 3.0GHz WS with RedHat 5. The time is not linear to the number of the exposures. Although the actual time it will take to reduce any dataset will differ, the times should provide an idea of the relative amount of time one can expect each step to take.

To save disk space, you can delete files after they have been used in the next step (except *.mos and *mflats*.fits, since they are also used in the standard object reduction). However, if there is any possibility that you will be re-reducing the data may be useful to keep the H*.fits (renamed), fTo_RH*.fits (flat fielded), and AspgfTo_RH*.fits (AG probe masked) files.

Note that other temporary files (tmp*; *.fits) are also produced in reduction processes. These can safely be ignored.

Reduction of target object

Step 1: Basic Data Check and Renaming of Data Frames

```
$ namechange.csh [raw fits file list]

  raw fits file list = names of raw data files
```

A useful step prior to data reduction is the renaming of the data files so that their name reflects the observation date, exposure, and component CCD.

The filename such as SUPA... is changed as H[Date][type][ID]_[chipname].fits.

The date YYMMDD is HST(Hawaii time), and 1 day before DATE-OBS.

ID is the frame serial number of the observation day of each type(bias,dark,object).It is very confusing but both target object and standard has "object" ID.

```
Example:

$ cd object/
```

enters into the directory of object frames

```
$ ls -1 SUPA*.fits > namechange.lis
```

The option of "ls -1" is "minus one".

```
$ namechange.csh namechange.lis
```

The namechange.lis should be like that

```
$ cat namechange.lis
SUPA00204610.fits
SUPA00204611.fits
SUPA00204612.fits
...
```

Result

The files are renamed as follows;

```
H030425object025_si001s.fits
H030425object025_si002s.fits
H030425object025_si005s.fits
...
```

If you make symbolic links to the files in ../spcam_training_data/, they still points to SUPA***, but their names are such as H030425object025_si001s.fits. That is OK

Note that each CCD of Suprime-Cam has a name:

```
-------- AG probe location ----------
w67c1 w6c1 si006s si002s w7c3
w93c2 w9c2 si005s si001s w4c5
```

Step 2: Overscan subtraction and bias subtraction

```
 $ overscansub.csh [overscansub.lis]

   overscansub.lis = list of raw data files
```

The script overscansub.csh issues commands that subtracts the median value of the overscan region in each line, and trims the overscan region off from the frame. The Suprime-Cam CCDs typically have an overscan level of about 10000 ADU.

The CCDs in Suprime-Cam have very little bias pattern, so only overscan is subtracted

```
Example:

$ ls -1 H*.fits > ovserscansub.lis
$ overscansub.csh ovserscansub.lis
```

The overscansub.lis should be like that

```
$ cat overscansub.lis
H030425object025_si001s.fits
H030425object025_si002s.fits
H030425object025_si005s.fits
...
```

Checkpoints:

Compare the values of original frame and overscan subtracted image. The latter should be about 10000 ADU smaller

Checking the file headers with a procedure like IRAF's imhead (cl> imhead H*.fits) should show that the image sizes are smaller after removing the overscan region.

Step 3: Make flat field frames

```
 $ mask_mkflat_HA.csh [mkflat.lis]  [base name]  [lower value]  [upper value]

   mkflat.lis = list of files to use to make flats
   base name = basename for the flats
   lower value = minimum value to accept (0.5 is recommended)
   upper value = maximum value to accept (1.3 is recommended)
```

The script mask_mkflat_HA.csh creates a flat from files with objects. The flat file is used to correct the difference of sensitivity of each pixel in a frame. Areas vignetted by the AG probe is masked out, normalized, and a median of them is taken.

There are three basic types of flats: sky flats (blank fields), twilight flats, and dome flats. Sky flats are thought to be the best since the optical path of the light and the spectral distribution of the light of the flat is identical to the actual target frames. In fact, the target frames can be used to produce flats as long as there are no large objects that extend several hundred pixels in the frames.

```
Example:

$ ls -1 To_RH*.fits >mkflat.lis
$ mask_mkflat_HA.csh mkflat.lis obj 0.5 1.3
```

This is an example of making a sky flat by combining the object frames

After running the script, there should be flat files for the 10 CCDs.

```
obj_mflat_si001s.fits
obj_mflat_si002s.fits
...
```

The mflat files should have values around unity and should have smooth pattern without much local structures. U-band and bands redder than z will have more structure than other bands. However, any local variations should be continuous. It there are sudden jumps in the flat values, the flat would be bad.

Note that the value -32768 is the blank value used in SDFRED, and it is OK.

Note 1:

In principle, a flat can be produced with a minimum of three exposures. However, the smaller the number of frames used, the larger the noise and residual effects of objects in the frame. We recommend to use at least 6 frames, ideally over 20 frames(exposures) to produce a flat (especially sky flats). In practice dataset, the flat is created from 5 exposures, and the error is large.

Note 2:

It is important not to mix the different types of flat frames since their background illumination have different slopes. For example, SDFRED may produce flats with discontinuous stripes when applied to frames with different illumination patterns. It is due to the algorithm used in SDFRED.

Note 3:

The SDFRED command uses a parameter file to mask out known bad columns and hot pixels. the default parameter file is optimized for data taken after August 2002 (Messia V). To reduce data taken before this, follow the procedures below.

```
$ cd sdfred20080610
```

For data taken before March 2001 (old CCD):

```
$ cp sdfredSH/mask_mkflat_HA/blankmap_oldCCD/* sdfredSH/mask_mkflat_HA/blankmap/
```

For data taken between April 2001 and August 2002 (Messia III):

```
$ cp sdfredSH/mask_mkflat_HA/blankmap_messiaIII/* sdfredSH/mask_mkflat_HA/blankmap/
```

To return to the default:

```
$ cp sdfredSH/mask_mkflat_HA/blankmap_messiaV/* sdfredSHmask_mkflat_HA/blankmap/
```

Note 4:

In the practice dataset, the flat is created in object/ directory, and used later in standard/ directory. Usually, several targets taken with the same filter is mixed to create flats. In such case, we can recommend to create another directory .../flat/ and make symbolic links of To_R* files in all targets, and execute the command in flat/ directory. (Note that the standard star frames that have shorter exposure times must not be included in the list.)

```
e.g.)
---(work directory root) - object1/
                         - object2/
                         - object3/
                         - flat/
                         - standard/
```

Step 4: Flat Fielding

```
 $ ffield.csh [ffiled_mf.lis]  [ffield_im.lis]

   ffield_mf.lis = list of flats to be used
   ffield_im.lis = list of (overscan subtracted) files to be flat fielded
```

This command corrects the pixel to pixel variation in sensitivity, and the effect of vignetting of the telescope optics.

```
Example:

$ ls -1 obj_mflat*.fits > ffield_mf.lis
$ ls -1 To_*.fits > ffield_im.lis
$ ffield.csh ffield_mf.lis ffield_im.lis
```

After the flat fielding, the background in each file should be almost flat. The circular illumination pattern seen in the raw data at the edge of the focal plane (w67c1, w93c2, w7c3, w4c5) should almost disappear. If the flat is poor, a slight (several %) illumination pattern may remain.

Step 5: Distortion correction and atmospheric dispersion correction

```
 $ distcorr.csh [distcorr.lis]

   distcorr.lis = list of (flat fielded) files to be corrected
```

The script distcorr.csh corrects the field distortion due to the telescope optics, and the differential atmospheric dispersion. The input frames are assumed to be flat-fielded images. The corrections are based on the airmass and other values recorded in the FITS header.

```
Example:
```

```
$ ls -1 fTo_RH030*.fits >distcorr.lis
$ distcorr.csh distcorr.lis
```

After the distortion correction, diagonal patterns may show up in the background. This is due to the fact that fractional pixel shifts smooth out the noise while integer pixel shifts leave the original noise characteristics intact.

Step 6: Measurement of the PSF size

```
 $ fwhmpsf_batch.csh [fwhmpsf_batch.lis] [max number of objects]
   [min peak flux] [max peak flux] [min FWHM] [max FWHM]

   fwhmpsf_batch.lis = list of images to check PSF
   max number of objects = the number of stars to use to measure
                           the PSF in each image
   min peak flux = minimum peak flux of stars to use
   max peak flux = maximum peak flux of stars to use
   min FWHM = minimum FWHM of stars to use
   max FWHM = maximum FWHM of stars to use
```

Before coadding, the equalization of PSF is needed. The script fwhmpsf_batch.csh is used to determine an appropriate target PSF for a list of images. The script measures the FWHM of the PSF in several images. The script outputs a log and a histogram to the standard output.

```
Example:

$ ls -1 gfTo_RH03042*.fits > fwhmpsf_batch.lis
$ fwhmpsf_batch.csh fwhmpsf_batch.lis 50 2000 30000 2.0 7.0 \
 > fwhmpsf_batch.log
```

The log file should contain entries such as:

```
gfTo_RH030425object025_si001s.fits   3.60   1 6 15 13 0
gfTo_RH030425object025_si002s.fits   3.80   1 1 20 16 0
gfTo_RH030425object025_si005s.fits   3.60   2 13 19 0 0
...
3.3 |
3.4 |
3.5 |**
3.6 |**
3.7 |*
3.8 |**
3.9 |**
4.0 |
4.1 |*
```

The log format of each line is as follows:

[name of image]

[mean FWHM of PSF]

[number of objects with within 0.1'' of mean PSF-0.2]

[# within 0.1'' of mean PSF-0.1]

[# within 0.1'' of mean PSF]

[# within 0.1'' of mean PSF+0.1]

[# within 0.1'' of mean PSF+0.2]

An ASCII histogram shows the distribution of mean PSFs in the images.

Some training is required to determine the target PSF size.

In the example above, the images have FWHM values centered around 3.7 pixels. Target=4.1 is the most conservative solution. If you are only interested in the flux of objects, but not the shape of the image, the conservative choice is recommended. Target=3.9 is also preferable, if you think the difference of 4.1 and 3.9 is negligible. Or, if there were plenty of data, you can exclude the images with the poor PSF (e.g, all data whose PSF is larger than 3.7), so that the PSF in the final image be better. The decision highly depends on the scientific goal.

Note 1:

```
 $ fwhmpsf.csh [image file] [max number of objects]
   [min peak flux] [max peak flux] [min FWHM] [max FWHM]

   image file = the image to check PSF
   max number of objects = the number of stars to use to measure
                           the PSF in each image
   min peak flux = minimum peak flux of stars to use
   max peak flux = maximum peak flux of stars to use
   min FWHM = minimum FWHM of stars to use
   max FWHM = maximum FWHM of stars to use
```

To find the PSF of a single image use the script fwhmpsf.csh. The parameters are the same as for fwhmpsf_batch.csh. Just feed it the name of a image rather than a list of images.

```
Example:
$ fwhmpsf.csh gfTo_RH030425object025_si001s.fits 50 2000 30000 2.0 7.0
```

This produces output like:

```
gfTo_RH030425object025_si001s.fits   3.60   1 6 15 13 0
```

This output indicates that the image gfTo_RH030425object025_si001s.fits has a PSF FWHM of 3.6 pixels.

Note 2:

```
 $ starselect.csh [image name][max number of objects][min peak flux]
     [max peak flux][min FWHM][max FWHM][output file]

   image name = name of image to check
   max number of objects = the number of stars to use to measure the PSF
   min peak flux = minimum peak flux of stars to use
   max peak flux = maximum peak flux of stars to use
   min FWHM = minimum FWHM of stars to use
   max FWHM = maximum FWHM of stars to use
   output file = name of file with location of selected stars
```

The script starselect.csh is useful for determining the appropriate parameters ([max number of objects] [min peak flux] [max peak flux] [min FWHM] and [max

FWHM]) for selecting stellar objects in an image.

```
$ starselect.csh gfTo_RH030425object025_si001s.fits 50 2000 \
  30000 2.0 7.0 output.reg
```

The script will produce an output file (output.erg) that contains the location of stellar objects that are selected using the given criteria. The output is formatted so the the image can be displayed with the locations of stars marked with a circle using saoimage-ds9. If the majority of objects selected are actual stellar objects, then the parameters are appropriate for psf_match for a given image. If the data quality varies from image to image, it is important to check whether or not a single set of parameters are appropriate for all the data. If not, it is better to run psfmatch_batch multiple times using the appropriate criteria for each subset of data.

Using saoimage-ds9 is the easiest way to display a image and overlay the location of the selected stars.

```
$ ds9 gfTo_RH030425object025_si001s.fits
```

Select Region, Load, and select output.reg. Then green circles will be overplotted on the image. If more than half of the objects selected are stellar objects, the parameters you adopted are appropriate

Note 3:

The scripts fwhmpsf_batch.psf, starselect.csh and psfmatch_batch.csh(next step), may not work in crowded fields. In crowded fields it may be necessary to estimate the PSF manually. In this case, to obtain the same results as the psfmatch script, each image that has a PSF more than 0.1'' smaller than the target PSF should be gaussian smoothed with a gaussian that has a sigma=sqrt(PSF_target^2 - PSF_image^2)/2.35482. The psfmatch program actually iterates around the given sigma in order to get the best matched final PSF.

Step 7: Equalize the PSF size

```
 $ psfmatch_batch.csh [psfmatch_batch.lis] [max number of objects]
     [min peak flux] [max peak flux] [min FWHM] [max FWHM] [target FWHM]

   psfmatch_batch.lis = the list of images to match to a single PSF
   max number of objects = the number of stars to use to measure the
                           PSF in each image
   min peak flux = minimum peak flux of stars to use
   max peak flux = maximum peak flux of stars to use
   min FWHM = minimum FWHM of stars to use
   max FWHM = maximum FWHM of stars to use
   target FWHM = FWHM to smooth all the data to
```

The script psfmatch_batch.csh matches the PSF of all images to be combined to a predetermined target FWHM. Images with PSFs smaller than the target (within a small range) are gaussian smoothed, other images are simply copied. The target PSF should represent the typical PSF of the worst exposure that is going to be combined.

The command prints a log to the standard output with the following columns:

```
[name of psf_matched_image]
[FWHM of PSF after matching]
[number of objects with within 0.1'' of target PSF-0.2]
[# within 0.1'' of  target PSF-0.1]
[# within 0.1'' of target PSF]
[# within 0.1'' of  target PSF+0.1]
[# within 0.1'' of  target PSF+0.2]
```

The log can be used to check if the PSF matching worked properly. If the command worked, then the number of objects with FWHM within 0.1'' of the target PSF should be largest, with fewer objects with smaller and larger FWHM. If this is not the case it is best to check the PSF by hand.

```
Example:

$ ls -1 gfTo_RH03042*.fits > psfmatch_batch.lis
$ psfmatch_batch.csh psfmatch_batch.lis 50 2000 30000 2.0 7.0 3.7 > psfmatch_batch.log
&
```

The command produces output like:

```
pgfTo_RH030429object017_si001s.fits   3.70   0 5 38 6 0
```

The IRAF procedure imexam is useful for checking PSFs. (Display image; cl> imexam image.fits; place cursor above a star; type "r" or "a" to measure FWHM)

Note that different softwares use different fitting algorithms and return different FWHM values. SDFRED uses FWHM values generated by SExtractor, which are different from those produced by IRAF's imexam task. The purpose of checking with IRAF is not for an exact match in the FWHM values, but to confirm that the output images have comparable PSF sizes after the matching.

The appropriate parameter values for psfmatch will change depending on the quality of the data. Different bandpasses, integration times, and weather conditions will require different parameters.

Step 8: Subtracting the Sky

```
 $ skysb.csh [skysb.lis] [sky-mesh]

   skysb.lis = list of images to sky subtract
   sky-mesh = size of mesh for determining sky values
```

The script skysb.csh computes a mesh pattern that represents the sky background and subtracts it from the image. The script determines the sky in sky-mesh x sky-mesh sized sections of the image and interpolates bilinearly. The mesh size must be at least twice the size of the largest object in interest.

```
Example:
$ ls -1 pgfTo_RH03042*.fits > skysb.lis
$ skysb.csh skysb.lis 64 > skysb.log
```

Once the sky is subtracted, the background in an image should be around 0 with no spatial slope. If there is a large object that occupies a large fraction of the image, the algorithm will fail. Crowded fields would also be problematic. The sky should be determined and subtracted manually in such cases

Step 9: Masking the AG Probe

```
 $ mask_AGX.csh [mask_AGX.lis]

   mask_AGX.lis = list of files to mask
```

The script mask_AGX.csh will mask areas vignetted by the AG probe by the value -32768. The script should only affect the top few hundred rows of the data from chips w671, w6c1, si006s, si002s, and w7c3. Other files are not affected.

```
Example:
$ ls -1 spgfTo_RH03042*.fits > mask_AGX.lis
$ mask_AGX.csh mask_AGX.lis
```

Although only half the CCDs are potentially affected by the AG probe, the input file list should include all the object files so that files with the same naming convention exist to make list making for subsequent steps easier.

Step 10: Masking Bad Pixels

Data in some pixels may be corrupted due to instrument trouble and other problems during the observations. Such regions need to be identified by inspecting individual data images and masked. For example, background areas that failed to flatten well and systematically deviate from 0 are candidates for masking. If enough images are stacked so that there is at least one image with good data contributing to all parts of a final combined image, there should be enough information for most analysis purposes, even though areas corresponding to masked data will have a lower signal to noise.

The SDFRED package contains three scripts that can be used to mask bad data.

Linear region

```
 $ line_bank [input image] [x1] [y1] [x2] [y2] [width]
    [blank value] [output image]

   input image = name of image to mask
   x1 = x coordinate of start of line
   y1 = y coordinate of start of line
   x2 = x coordinate of end of line
   y2 = y coordinate of end of line
   width = width of line
   blank value = mask value (usually -32768)
   output image = name of masked image
```

The script line_bank masks a linear structure such as satellite trails.

```
Example:

$ line_blank AspgfTo_RH030425object025_si001s.fits \
  88 112 1940 837 30 -32768 lAspgfTo_RH030425object025_si001s.fits
```

The example masks line which crosses (88,112) and (1940,837) and around 30 pixels width.

Circular region

```
 $ circular_blanks [input image] [blanklist] [blank value] [output image]

   input image = name of image to mask
   blank list = file that contains the x and y coordinates and
                the radius of the area to be masked
   blank value = mask value (usually -32768)
   output image = name of masked image
```

The script circular_blanks masks circular regions.

```
Example:

$ circular_blanks lAspgfTo_RH030425object025_si001.fits \
  blanklist -32768 clAspgfTo_RH030425object025_si001s.fits
```

where blanklist looks like:

```
$ cat blanklist
365 1835 80
1202 3582 100
```

The two lines correspond to circle of (x,y,r)=(365,1835,80) and (x,y,r)=(1202,3582,100).

Rectangular regions

```
 $ blank.csh [blank list]

   blank list = list of files with  rectangular regions that have
                to be masked.
```

For each image, xxx.fits, in the blank list the script blank.csh will look for a file named as blank_xxx in the same directory, and mask rectangular regions specified in the file to -32768. Each line in the file blank_xxx should contain the x and y coordinates of two opposite corners of a rectangular area.

The IRAF routine imexam is useful for getting the coordinates. (cl> imexam; press "b" at two corners to define a rectangle; the coordinates of the corners will be printed to the screen in the order of x1 x2 y1 y2.)

```
Example:

$ ls -1 AspgfTo_RH03042*.fits > blank.lis
$ blank.csh blank.lis
```

Mask files have been included for a subset of images,

blankmap_AspgfTo_RH030425object025_si002s

...

These files have entries like:

```
$ cat blankmap_AspgfTo_RH030425object025_si002s
1974 2034 2356 2634
1528 1804 4024 4070
...
```

The script masks the regions specified for files with mask files and will simply create copy files that don't need to be masked. Often, if the flat is poor, the same region of each image corresponding to a particular CCD may need to be masked.

Note that the blankmap_* parameter files included in the practice dataset is from the Subaru Deep Field project and are not necessarily suitable for the practice dataset. If all of the sample files are applied to the practice data, areas that shouldn't normally be masked will be masked. These files are strictly for practice use. You may wish to create your own mask files to apply to the practice data.

Step 11: Determine Alignment and Scaling

```
 $ makemos.csh [makemos.lis] [starsel nskysigma] [starsel npix]
              [starsel peakmin] [starsel peakmax]
              [aperture phot radius in pix] [output mos-file name]

   makemos.lis = list of images to align
   starsel nskysigma = signal to noise level of objects
                       to use for alignment
   starsel npix = number of continuous pixels with [starsel nskysigma]
                  to identify object
   starsel peakmin = minimum value of peak pixel of alignment stars
   starsel peakmax = maximum value of peak pixel of alignment stars
   aperture phot radius in pix = radius to use for aperture photometry
   output mos-file name = file to record alignment and scaling
```

Signal to noise can be improved by combining multiple images to produce a final image. The script makemos.csh determines the relative shifts, rotations, and flux scales of different images. The script identifies stellar objects in each image and determines the the shifts, rotations, and flux scale from stellar objects common to multiple images. The first image in the list is used as the reference image.

```
Example:

$ ls -1 bAspgfTo_RH03042*.fits > makemos.lis
$ makemos.csh makemos.lis 5 20 500 10000 10 all.mos > makemos.log
```

The script will print to the standard output the number of stellar objects selected for alignment and scaling.

```
     ...
     selected stars = 721
     ...
```

The script is likely to fail if the number of selected stars per image is small (< 30) or very large. Adjust the parameters [starsel nskysigma], [starsel npix], [starsel peakmin], and [starsel peakmax] will help the script select appropriate stellar objects.

The best parameters for selecting objects in this step is different from PSF measurement, because a different underlying algorithm is used to find a wider range of objects to determine relative positions and flux scaling that works over a range of fluxes.

```
Example:

$ cat all.mos
bAspgfTo_RH030425object025_si001s.fits 0.000000 0.000000 0.000000 1.000000
bAspgfTo_RH030425object025_si002s.fits 1.601005 4088.551744 -0.000275 0.989406
```

```
bAspgfTo_RH030425object025_si005s.fits -2118.787371 1.104977 -0.000282 0.983195
...
```

There are five columns in the output *.mos file: the name of the image, the x offset, the y offset, the counter clockwise rotation (radian), and flux ratio. If all rows have values in the five columns, the alignment and scaling was probably successful. If the alignment and scaling are unsuccessful, the output file may not even be produced or may have missing or unreasonable values.

The procedure for quick check of all.mos is still under discussion. Some examples are as follows:

1. Examining the final image created in the next step is one of the most important checkpoint. However, it is not the only point. If the number of exposure is large, it is difficult to detect some small defects by eye inspection from the final image.
2. Make a scatter plot of 2nd and 3rd column of all.mos. The result shows the relative position of each shot, and represent the dither pattern and the chip position. If there is a large leap, the matching would have been failed.
3. The distance between chips should be almost constant. (A slight change would exist due to atmospheric dispersion between chips.) The distance of two chips of the same exposure, for example between si001s and w67c1, is largely different from that of other exposures, the data of the corresponding shot would be incorrect.
4. The fifth column of *.mos (relative flux) of a chip should be almost proportional to the exposure time, if sky condition is photometric. (It is affected by atmospheric extinction (airmass) though.)

In the next step, each image is converted with the data in all.mos as follows;

```
x_mos =   cos(theta) x  -sin(theta) y + x_local
y_mos =   sin(theta) x  +cos(theta) y + y_local
```

Note 1:

If combining is not needed, you may skip Step 11 and 12, and end the reduction. Or, if the data is not contiguous, you cannot succeed in this step.

Step 12: Combining

```
 $ imcio2a [parameters] [mos file] [result image]

 parameters = parameters that define the combining algorithm usually
  "-dist_clip -nline=20 -dtype=FITSFLOAT -pixignr=-32768"
 mos file = file containing the alignment and scaling values
            (output from makemos.csh)
 result image= the name of the final image
```

imcio2a combines the images into a final combined image using the output from makemos.csh (*.mos). Using the parameter -dist_clip will combine the images using a clipped mean algorithm.

```
Example:

$ imcio2a -dist_clip -nline=20 -dtype=FITSFLOAT -pixignr=-32768 all.mos all.fits
```

The parameter -dist_clip can be replaced by -dist_med to get a weighted median combined image or -dist_add to use a weighted mean pixel values.

Note that the header of the new image is incomplete. Use the first file listed in makemos.lis in the previous step as a reference header.

Here are the meanings of the typical parameters:

-dist_clip : use a clipped mean algorithm for combining

-nline=20 : set the y direction buffer width to 20

-dtype=FITSFLOAT : make the output data floating point

-pixignr=-32768 : ignore pixels valued -32768

For details and other optional parameters of imcio2a will be printed by

```
 $ imcio2a -h
```

Reduction of standard object

Following 4 steps is for data reduction of standard objects. They are the same as that of target object, except the procedure is done in standard/ directory, and the flat frames are simply copied from object/ directory.

Step 1S Renaming

```
$ namechange.csh [raw fits file list]

  raw fits file list = names of raw data files
```

Renaming is done in the standard/ directory

```
Example:

$ cd standard/
```

enters into the directory of standard frames

```
$ ls -1 SUPA*.fits > namechange.lis
$ namechange.csh namechange.lis
```

The namechange.lis should be like that

```
$ cat namechange.lis
SUPA00195120.fits
SUPA00195121.fits
SUPA00195122.fits
...
```

Result

The files are renamed as follows;

```
H030330object044_si001s.fits
H030330object044_si002s.fits
H030330object044_si005s.fits
...
```

Step 2S Overscan and bias subtraction

```
 $ overscansub.csh [overscansub.lis]

   overscansub.lis = list of raw data files
```

In standard/ directory, overscan is subtracted from all the data.

```
Example:

$ ls -1 H*.fits > overscansub.lis
$ overscansub.csh overscansub.lis

$ cat overscansub.lis
H030330object044_si001s.fits
H030330object044_si002s.fits
H030330object044_si005s.fits
...
```

and, To_RH030330object044_si001s.fits ... are created.

Step 3S Flat fielding

```
 $ ffield.csh [ffiled_mf.lis]  [ffield_im.lis]

   ffield_mf.lis = list of flats to be used
   ffield_im.lis = list of (overscan subtracted) files to be flat fielded
```

The purpose of standard object frames is to calibrate target frames. Therefore the flat frames must be the same as that used in the reduction of target object.

```
Example:

$ cp ../object/obj_mflat*.fits .
$ ls -1 obj_mflat*.fits > ffield_mf.lis
$ ls -1 To_RH*.fits > ffield_im.lis
$ ffield.csh ffield_mf.lis ffield_im.lis
```

and fTo_RH030330object044_si001s.fits ... are created.

Step 4S Distortion correction and atmospheric dispersion correction

```
 $ distcorr.csh [distcorr.lis]

   distcorr.lis = list of (flat fielded) files to be corrected
```

The distortion correction is required since it slightly changes the size of pixels, and therefore flux.

```
Example:

$ ls -1 fTo_RH*.fits >distcorr.lis
$ distcorr.csh distcorr.lis
```

Step 5S Correction of relative flux scale among chips

After step (4S), the relative flux between different chips is not corrected. For example, the sensitivity of w67c1 is about half in 2002-2008/06 data. If there is a star which has 10000 ADU in w67c1, it should have ~ 20000 ADU if it were observed in other chip. The relative flux scale should be corrected according to *.mos created in step (11) of target object.

This step is unnecessary if standard star is only in si001s. Since standard stars are distributed in several chips in the practice data, this process is needed.

In this step, the data is divided by a typical relative value of the chip to the reference chip. Currently (SDFRED ver1.*), the script for this step is not provided. Users should do this process manually.

Notices on Data

In principle, SDFRED (ver1.*) can reduce Suprime-Cam data taken between 2000/10 and 2008/07. However, data taken before 2001/04 have several problems.

1. Data in 2000/11 - 2000/12
   The CCD linearity has problem for DET-ID=0,1,7,8,9.
2. Data in 2000/11 - 2001/01
   The WCS information in the FITS header is wrong. The flag of CDELT1 is negated, and the CD matrix is wrong. Since SDFRED uses WCS information in step (11), registration (makemos.csh) fails with this data.
3. Data in 2000/10 - 2001/03
   One CCD (DET-ID=6; w67c1) is dead. However, reduction with SDFRED is possible for 9 chips. Note that CCDs were replaced in 2001/04 and flat pattern and gain is changed, while the basic format of CCDs (e.g., overscan format, chip names) is unchanged.
4. Data in 2008/07/21 - now
   Suprime-Cam is largely upgraded in 2008/07, including CCD replacement and change of file format. SDFRED ver1.* cannot reduce the data.

Acknowledgment

SDFRED team thanks Takehiko Wada (JAXA), Chiaki Ihara (JAXA), Hitoshi Hanami (Iwate Univ), Myungkook James Jee (JHU), Kazuaki Ota (NAOJ), Yasunori Sato (NAOJ), Ryosuke Yamauchi (Tohoku Univ), Elinor Medezinski (Tel-Aviv Univ), Dovi Poznanski (Tel-Aviv Univ), Ben Cain (MIT), Tomoki Saito (Ehime Univ), Sakurako Okamoto (Univ of Tokyo), Alice Shapley(UCLA) and Naoki Yasuda (Univ of Tokyo) for their comments and bug report.