Suprime-Cam Data Reduction Textbook with SDFRED2

2014 Subaru Winter School at KASI

February 25–27, 2014

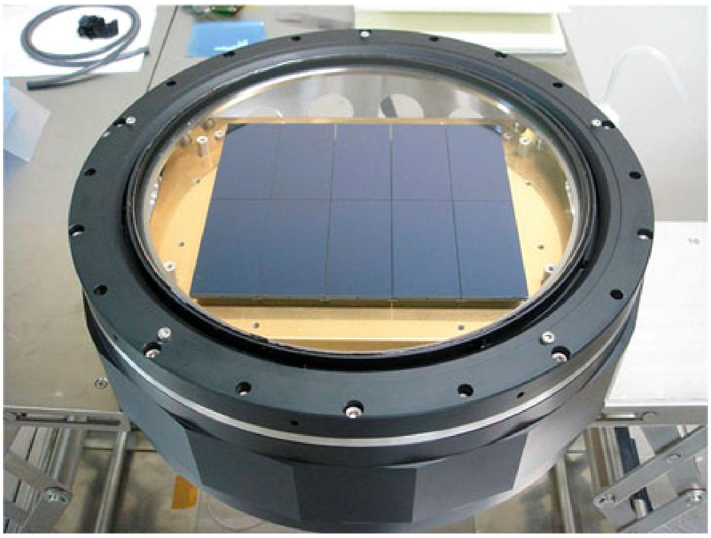
! Preliminary draft !

Foreword

This textbook describes a typical procedure using the SDFRED2 software package to reduce optical imaging data taken with the Subaru Prime Focus Camera (Suprime-Cam) mounted on Subaru Telescope. SDFRED2 was developed based on SDFRED1 (formerly SDFRED), which was originally developed by Drs. M. Ouchi and M. Yagi for reducing Suprime-Cam data after the installation of the new CCD chips in July 2008. The copyright for SDFRED2 software belongs to Masami Ouchi. You may freely copy and distribute SDFRED2, but the paper written by the author (Ouchi et al., 2004, ApJ, 611, 660) should be cited in any scientific paper based on data reduced with SDFRED2.

This document is a re-edited version of the SDFRED2 manual[1] written by Drs. F. Nakata. and R. S. Furuya.

T. Terai (Subaru Support Astronomer)
E-mail : tsuyoshi.terai“at”nao.ac.jp



Suprime-Cam’s focal plain

The cover image — Composite tricolor (B, R, $IA651$) images of NGC2403 taken on October 13, 2005 (`http://subarutelescope.org/Gallery/pressrelease.html`).

[1]`http://www.naoj.org/Observing/Instruments/SCam/sdfred/v2.0/sdfred2_2p1ae.pdf`

Contents

1 Introduction

SDFRED2, the Suprime-Cam Deep Field REDuction package 2, allows users to reduce data taken with Subaru Prime Focus Camera (Suprime-Cam) which is a mosaic CCD camera with ten 2048 × 4096-pixels CCDs. SDFRED2 allows the immense data output from Suprime-Cam to be reduced semi-automatically, or even fully automatically, using standard parameters. The resulting reduced images are then ready for scientific analysis.

SDFRED2 is aimed for photometry of deep-field or blank-field imaging. Therefore, SDFRED2 may not be able to properly reduce images that contain object(s) spread over a significant portion of a chip. Moreover, we have not performed any quantitative tests for shape measurements to images produced by SDFRED2, such as those for weak lensing analysis. Special procedures and cautions for reducing such data will be described in each section.

```
SDFRED2 is designed to reduce data taken as of 2008 July 21.

All the data taken on and before 2008 July 20
MUST be reduced using SDFRED1.
```

Suprime-Cam was largely upgraded in July 2008, including CCD replacement and modification of the file format. Special cautions is indicated for reducing some data, as described in Appendix A. Check there to see if this applies to your data.

2 Getting Started

2.1 Computer Hardware and Operating System Requirements

SDFRED2 requires a UNIX-based computer equipped with 256MB or more of memory. Requirements for storage space depend on the quantity of data, but typically you need several GB to several hundred GB (20 GB will sufficient to reduce the training data set).

The SDFRED2 software has been developed and tested on Linux machines of CPU type X86_64, kernel version 2.6.18–194.17.1.el5, glibc: version 2.5 and gcc version of 4.1.2. as of 2011 January 14.

2.2 Getting the SDFRED2 Package

The latest version of the package can be obtained from `http://subarutelescope.org/Observing/Instruments/SCam/sdfred/sdfred2.html.en`, and we recommend that you visit this web page from time to time for updated information[2].

[2] A summary of possible problems in Suprime-Cam data is given in the Subaru Mitaka Okayama Kiso Archive system (SMOKA) web page (http://smoka.nao.ac.jp/about/subaru.jsp). The web page for the notice of the data before June 2008 (http://smoka.nao.ac.jp/help/help supdetailNEW.jsp) may also help

2.3 Other Software Requirements

In addition to SDFRED2, we recommend the following software packages. Please note that the first two are "must-have" packages; the last one is optional:

- Basic software packages: C compiler, Perl, csh, sh/bash
- SExtractor. Find at `http://www.astromatic.net/software/sextractor` (Bertin & Arnouts 1996, A&AS, 117, 393)
- IRAF. Find at `http://iraf.nao.ac.jp/` or `http://iraf.net/`, this is not necessarily required for SDFRED2, but is pretty handy to have. The latest version of SFRED2 (December 27, 2010) has been tested with version IRAF 2.14.1.

2.4 Installation

2.4.1 Uncompressing the software package

Uncompress the download package by,

```
$ tar xvzf sdfred20130924_mf2.tar.gz
  or
$ gunzip -c sdfred20130924_mf2.tar.gz | tar xvf -
```

2.4.2 Compilation

Go into (`cd`) the `sdfred20130924_mf2` directory and build the software as follows.

```
$ cd sdfred20130924_mf2/
$ ./configure
$ make all
```

The programs are now installed into `sdfred20130924_mf2/bin/`.

2.4.3 Adding the directory to the PATH

Use an editor to add the `sdfred20130924_mf2/bin` directory (executables directory) to your PATH environment in your shell configuration file. You should modify the path to work with your environment.

bash users: Edit ~/.bashrc and ~/.bash_profile as follows;

you to get a hint for resolving your questions.

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

```
  PATH=[path to sdfred20130924_mf2]/sdfred20130924_mf2/bin:$PATH
  export PATH
```

~/.bashrc is used when a new terminal is open, ~/.bash_profile is used when you log into the computer. Therefore, you need to modify both files. After the files are updated, adjust 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 SDFRED2 binaries are installed (executables directory) as follows;

```
Example:
  % emacs ~/.cshrc
  % rehash
```

```
  $ set path=([path to sdfred20130924_mf2]/sdfred20130924_mf2/bin
             $path)
```

2.4.4 Setting 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;

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

Add the 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 the following two lines.

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

and reflect the setting to the current session.

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

2.4.5 Other settings

If you prefer to use IRAF, don't forget to execute `mkiraf` in your "work directory". Additional settings may be required, depending on your computer environment. When you are finished with the settings, make sure your environment is as follows

```
Example:
  $ env
  $ which namechange.csh
```

In `env` command result, check LANG, and LC_ALL environment. Make sure the "which" command can find the directory you have installed. Once you have successfully finished the software preparation, you do not have to take these steps again.

2.5 Preparation of Data

Before starting a reduction, you need to sort data files into directories. SDFRED2 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 another way.

Sample Data A sample dataset is available at
`http://subarutelescope.org/Observing/Instruments/SCam/sdfred/data/spcam_training_data_fdccd_1.tar.gz` (590MB)
`http://subarutelescope.org/Observing/Instruments/SCam/sdfred/data/spcam_training_data_fdccd_2.tar.gz` (570MB)

Notice that both the data sets must be downloaded.

We have checked that the software works well for this dataset. If you encounter any problem(s) during regular data reduction, we suggest you diagnose it using this well-tested sample data. The specific examples given in this manual refer to this sample dataset.

```
Example: extracting sample data
  $ tar xvzf spcam_training_data_fdccd_1.tar.gz
  $ tar xvzf spcam_training_data_fdccd_2.tar.gz
```

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

Moreover, inspect images by eye using saoimage-ds9 or your favorite FITS viewer. This is a good starting point to check for such issues as whether there are any files (i.e. exposures) showing distorted/elongated stars, and allows you to eliminate bad data. SUPA010998*.fits, and SUPA010999*.fits are the object frames (target object), SUPA0109971*.fits are the standard frames, and SUPA01102*.fits are dome-flat frames in the sample dataset.

In this version (Ver.2.0), both the data sets are assumed to be reduced/analyzed in three subdirectories (`object/ standard/ flat/`) which are in the same directory as `spcam_training_data_fdccd`.

```
---(work directory root) - spcam_training_data_fdccd/
                         - object/
                         - standard/
                         - flat/
```

An example of making symbolic links:

```
  $ mkdir object standard flat
```

The result of using `ls` is;

```
  $ ls -1
  object
  spcam_training_data_fdccd
  spcam_training_data_fdccd_1.tar.gz
  spcam_training_data_fdccd_2.tar.gz
  standard
```

Notice that the option of "`ls -1`" is "minus one".

Then link object frames into `object/` directory

```
$ cd object/
$ ln -s ../spcam_training_data_fdccd/SUPA010998*.fits .
$ ln -s ../spcam_training_data_fdccd/SUPA010999*.fits .
```

Copy blank map data

```
$ cp ../spcam_training_data_fdccd/blankmap* .
$ cp ../spcam_training_data_fdccd/lblank.txt .
```

Link standard object frames into `standard/` directory

```
$ cd ../standard/
$ ln -s ../spcam_training_data_fdccd/SUPA0109971*.fits .
```

Link dome flat frames into `flat/` directory

```
$ cd ../flat/
$ ln -s ../spcam_training_data_fdccd/SUPA011002*.fits .
```

Check that 10 FITS files are in `standard/` directory, 30 FITS files are in `flat/` directory, and 50 FITS files and two blankmaps are in `object/` directory. In this textbook, we assume that readers will work on making flat frames, reducing the scientific targets, and reducing standard stars in each subdirectory created in above.

Retrieval from SMOKA All data obtained with the Subaru Telescope can be retrieved through the archive system, SMOKA (`http://smoka.nao.ac.jp`), after the 18-months of the proprietary period has passed. After getting the desired data sets, we suggest moving data of the target object and that of the standard object into separate directories. Check that all the observations catalogued in a directory were taken with the same filter (e.g V-band), and are the same data type (`object/standard`).

3 Data Reduction: Overview

3.1 A Typical Procedure

In this subsection, we provide an overview of data reduction using SDFRED2. The typical procedure for reducing target object frames and for standard object frames, respectively, is summarized in Tables 1 and 2, and details are described in §4 and §5.

Each of the processes applies to a different subset of data frames. The basic data flow consists of making lists of these various subsets, and then providing these lists to various programs within the SDFRED2 package. Each step produces a new set of images or data files. The naming convention for the various new files are in parentheses after each step in the list above.

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). Users are advised to keep the `H*.fits` (renamed), `fTo_RH*.fits` (flat fielded), and `AspgfTo_RH*.fits` (AG probe masked) files, since you never know when you might need to re-reduce these data sets. Note that other temporary files (`tmp*`; `*.fits`) are also produced during the reduction processes. These can be removed.

All the input files used in each step must exist under the current directory. If you want to process any files that aren't in the current directory, you have to tell the program where they are located. For example, make symbolic links to them, like

```
% ln -s [path to the data directory]/*.fits .
```

Table 1: Outline of Reducing Target Object Frames

Step	Purpose	Command	Files Generated
(1)	Renaming	`namechange.csh`	H*.fits
(2)	Overscan subtraction	`overscansub.csh`	To RH*.fits
(3)	Making flat frames	`mask_mkflat_HA.csh`	*mflats*.fits
(4)	Flat fielding	`ffield.csh`	fTo_RH*.fits
(5)	Distortion correction and atmospheric dispersion correction	`distcorr.csh`	gfTo_RH*.fits
(6)	PSF size measurement	`fwhmpsf_batch.csh`	—
(7)	PSF size equalization	`psfmatch_batch.csh`	pgfTo_RH*.fits
(8)	Sky subtraction	`skysb.csh`	spgfTo_RH*.fits
(9)	Masking out AG probe	`mask_AGX.csh`	AspgfTo_RH*.fits
(10)	Masking out bad regions	`blank.csh, ....`	bAspgfTo_RH*.fits
(11)	Alignment	`makemos.csh`	*.mos
(12)	Coadding	`imcio2a`	Final Image

Table 2: Outline of Reducing Standard Object Data

Step	Purpose	Command	Files Generated
(1S)	Renaming	`namechange.csh`	H*.fits
(2S)	Overscan subtraction	`overscansub.csh`	To_RH*.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	---	

4 Reducing Target Object Data

4.1 Initial Data Inspection and Renaming of Data Frames (Step 1)

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

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

Before reducing your data, rename the data files using information such as the date of observations, exposure, and component CCD. A filename such as SUPA... thus becomes H[Date][type][ID] [chipname].fits, where the date, YYMMDD, is one day prior to DATE-OBS (UT), and corresponds to Hawaiian Standard Time (HST) of the first half of the night. ID is the frame serial number of the observation day of each type (bias, dark, object). It should be noted that both target object(s) and standard star(s) have the same ID of "object".

```
Example:
  $ cd object/
```

enters the directory of object frames

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

The namechange.lis should be like that

```
$ cat namechange.lis
SUPA01099880.fits
SUPA01099881.fits
SUPA01099882.fits
  ...
```

Also rename flat frame data as

```
% cd ../flat
% ls -1 SUPA*.fits > namechange.lis
% namechange.csh namechange.lis
```

After the routine executes, your file names with SUPA ... should be changed as follows, under the `/object` directory

`H090523object038_chihiro.fits`
`H090523object038_clarisse.fits`
`H090523object038_fio.fits`
...

and, under the `/flat` directory

`H090523object077_chihiro.fits`
`H090523object077_clarisse.fits`
`H090523object077_fio.fits`
...

If you make symbolic links to the files in a separate directory, they still point to SUPA∗∗∗, but their names show up as `H090523object038_chihiro.fits`. That is okay.

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

[AG probe location]

6 *chihiro*	7 *clarisse*	2 *fio*	1 *kiki*	0 *nausicaa*
8 *ponyo*	9 *san*	5 *satsuki*	4 *sheeta*	3 *sophie*

Notice that if you defined an alias of "`ls`" as "`ls -F`", the command of "`ls -1 SUPA*.fits > namechange.lis`" will add @marks to the end of file names. Note that this does not happen if you use the command "`cat namechange.lis`".

4.2 Subtraction Overscan and Bias (Step 2)

The script **overscansub.csh** issues a command that subtracts the median value of the overscan region in each line, and trims the overscan region from the frame. Bias will be subtracted by assuming that the bias value is equal to that of the overscan. First, the script subtracts the median of the serial overscanned regions located at the right- or left-edges of the CCDs from each column of the pixel array. Second, the bias subtraction will be completed by subtracting medians for the individual parallel overscanned regions that are located at the top- or bottomedges of the CCDs.

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

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

Subtracting Overscan The Suprime-Cam CCDs typically have an overscan level of about 100–300 ADU.

Since the CCDs in Suprime-Cam have very little bias pattern, our experience suggests that subtracting overscan should suffice for many cases.

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

Makes a list file of the images to be used for the analysis.

```
% cd ../object
% ls -1 H090*.fits > overscansub.lis
```

Executes the "overscansub.csh" script.

```
% overscansub.csh overscansub.lis
```

Subtracts bias from the flat data as well by the following:

```
% cd ../flat
% ls -1 H090*.fits > overscansub.lis
% overscansub.csh overscansub.lis
```

After the execution, the overscan and bias subtracted images should be as follows,

```
To_RH090523object038_chihiro.fits
To_RH090523object038_clarisse.fits
To_RH090523object038_fio.fits
...
```

Checkpoints

- Compare the count statistics (e.g., average and/or median) for any regions where no objects are detected (the background) between the original frames and overscan subtracted image(s). The latter should be approximately 100–300 ADU smaller than the original. Note that the counts to be subtracted may be different in the individual CCDs and/or pixels.

- Check that the sizes of the output images (i.e., overscan-subtracted) are smaller than those of the input files. This can be checked with e.g., task `imhead` in IRAF by `cl> imhead H*.fits`.

4.3 Making Flat Field Frames (Step 3)

```
$ 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 (0.4 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 in sensitivities between pixels in a frame. Areas vignetted by the auto-guider (AG) probe are masked out, normalized and a median of all such areas is taken.

In general, there are three basic types of flats: object flats (blank fields), twilight flats, and dome flats. Object flats usually give the best result. In fact, the target frames can be used to produce flats as long as there are no large objects that extend (spread) several hundred pixels in the frames. In the specific case of the training data, we are dealing with the relatively nearby galaxy cluster, Abell 1689, where several bright objects exist at the center of the cluster. We do not suggest making flat frame(s) from these particular data. In this case, we suggest using dome-flat data. When working with this data, make a flat frame under the /flat directory and use it for the analysis of the standard stars as well.

```
Example:
  $ cd ../flat
  $ ls -1 To_RH090*.fits > mkflat.lis
  $ mask_mkflat_HA.csh mkflat.lis dome 0.4 1.3
```

The first command is to move to the directory where you stored the data. 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.

```
dome_mflat_chihiro.fits
dome_mflat_clarisse.fits
 ...
```

The `mflat` files should have values around unity and should have a smooth pattern without much local structure. The U-band data and any bands redward of z will have more structure than other bands. However, any local variations should be continuous. If there are abrupt changes in the flat values, consider creating another flat after eliminating (possible) bad exposures. Note that the value of -32768 is given for any blanked pixels by SDFRED2, and is not an error. However, if you identify such -32768 values over a large area or/and many pixels, the program may have failed in flat fielding. In this case, we strongly suggest checking the input parameters such as `[lower_value]` and/or `[upper_value]`.

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 using at least six frames, ideally over 20 frames, to produce a sensible flat — especially if you attempt to make it from sky frames.

Note 2: keep in mind that users should not mix the different types of flat exposures to make a flat frame. This is because the background illuminations have intrinsically different slopes. For example, SDFRED2 may produce flats with discontinuous stripes when applied to frames with different illumination patterns. This is due to the algorithm used in SDFRED2.

Note 3: the SDFRED2 command uses a parameter file to mask out known bad columns and hot pixels.

4.4 Flat Fielding (Step 4)

```
$ ffield.csh [ffield_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 pixel-to-pixel variation in sensitivity, and the effect of vignetting of the telescope optics.

```
Example:
  $ ls -1 dome_mflat*.fits > ffield_mf.lis
  $ ls -1 To_*.fits > ffield_im.lis
  $ ffield.csh ffield_mf.lis ffield_im.lis
```

Takes you to the directory where you stored your data.

```
  % cd ../object
```

Links the flat frame made at the **Step 3** to the object directory,

```
  % ln -s ../flat/dome_mflat*.fits .
```

Make a list file for the flat frames produced in the §4.3.

```
  % ls -1 dome_mflat*.fits > ffield_mf.lis
```

Make a list file for the images to be applied the flat fielding.

```
  % ls -1 To_RH090*.fits > ffield_im.lis
```

Executes `ffield.csh`.

```
  % ffield.csh ffield_mf.lis ffield_im.lis
```

After 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 (*chihiro*, *nausicca*, *ponyo*, and *sophie*) should have disappeared. Check to see if there is a low-level (several percent of variation) illumination pattern.

4.5 Distortion Correction and Atmospheric Dispersion Correction (Step 5)

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

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

The script `distcorr.csh` corrects the field distortion due to 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
```

Although the amount of distortion correction should be a function of the wavelength, we adopt measurements with the MIT CCD (which had been used till June 2008) at the R-band for SDFRED2. Please note that SDFRED2 does not check/optimize such parameters for the data taken as of July 2008 using the newly installed CCDs (FDCCD).

4.6 Measurement of PSF sizes (Step 6)

```
$ 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, equalization of the PSF is required. The script `fwhmpsf_batch.csh` is used to determine an appropriate target PSF for the images. The script measures the FWHM of the point-like sources (stellar objects) to obtain PSF sizes in several images, and outputs the results to the terminal, exposure by exposure, as shown below.

```
Example:
  $ ls -1 gfTo_RH090*.fits > fwhmpsf_batch.lis
  $ fwhmpsf_batch.csh fwhmpsf_batch.lis 50 2000 40000 2.0 7.0
```

The command produces output like:

```
gfTo_RH090523object038_chihiro.fits 4.10 1 5 12 0 0
gfTo_RH090523object038_clarisse.fits 4.00 0 3 16 18 0
gfTo_RH090523object038_fio.fits 4.10 1 11 21 0 0
  ...
3.5 |****
3.6 |*
3.7 |******
3.8 |**
```

```
3.9 |*
4.0 |******
4.1 |*******
4.2 |******
4.3 |*
```

To judge whether or not the PSF matching has ended successfully, you should check the following two points:

(1) Checking PSF sizes in each frame — Check the results shown by ascii text as follows:
`gfTo_RH090523object038_chihiro.fits 4.10 1 5 12 0 0`. This gives results of the 20 PSF measurement for each CCD chip at a single exposure. Individual columns show the following information:

1st : Name of image
2nd : Mean FWHM of PSF after PSF matching in pixel unit
3rd : Number of objects having PSF sizes within 0.1 pixels centered on (mean PSF − 0.2 pixel)
4th : Number of objects having PSF sizes within 0.1 pixels centered on (mean PSF − 0.1 pixel)
5th : Number of objects having PSF sizes within 0.1 pixels centered on mean PSF
6th : Number of objects having PSF sizes within 0.1 pixels centered on (mean PSF + 0.1 pixel)
7th : Number of objects having PSF sizes within 0.1 pixels centered on (mean PSF + 0.2 pixel)

Make sure that the number of objects falling into the bin including the mean, the column 5, shows the largest number of the distribution[3]. The bins of ±0.1 pixel, the column 4 and column 6, should be the next largest ones.

(2) Checking the overall results of the PSF measurements — Next, check the histogram presented by ∗ marks. This histogram summarizes your PSF measurements over the ten CCDs and all the exposures. In this particular example, one of the asterisks at PSF = 4.1 is from "`gfTo_RH090523object038_chihiro.fits 4.10 1 5 12 0 0`". Using all the information above, you need to select an appropriate PSF size, and supply it at `Step 7` (§4.7). Since selecting a PSF size strongly depends on your scientific goals, verify the results carefully. If needed, identify any exposure(s) whose PSF sizes are degraded, and exclude such exposure(s).

If you want to check the results with other software, the IRAF task "`imexam`" is handy for checking PSFs (display image; `cl> imexam image.fits`; place cursor above a star; type "`r`" or "`a`" to measure FWHM). Keep in mind that each software routine may be using different fitting algorithms and may return different FWHM values. SDFRED2 adopts FWHM values generated by `SExtractor`, which are different from those produced

[3]In general, one can expect that the mean value of PSF sizes measured in an exposure should peak in its histogram, if one measures PSFs using point-like sources only.

by IRAF's `imexam` task. The purpose of checking with IRAF is not to find an exact match in FWHM values, but to confirm that the output images have comparable PSF sizes after the matching.

```
$ 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
```

Note 1: Use the script `fwhmpsf.csh` to find the PSF of a single image. The parameters are the same as for `fwhmpsf_batch.csh`. Just supply the name of an image rather than a list of images.

```
Example:
  $ fwhmpsf.csh gfTo_RH090523object038_chihiro.fits 50 2000 40000
    2.0 7.0
```

This produces output like:

`gfTo_RH090523object038_chihiro.fits 4.10 1 5 12 0 0`

This output indicates that the image `gfTo_RH090523object038_chihiro.fits` has a PSF FWHM of 4.1 pixels.

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

   image = 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
```

Note 2: This script is designated to search for 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_RH090523object038_chihiro.fits 50 2000 \
  40000 2.0 7.0 output.reg
```

The script will produce an output file (output.reg) that contains the location of stellar objects satisfying the given criteria. The output is formatted so that the stellar objects are plotted with green circles when you plot using `saoimage-ds9`. If the majority of the selected objects are "real stellar objects" (stars in many cases), then the parameters are appropriate for `psf_match` for a given image. If you realize that the quality of the data varies image by image, determine whether or not a single set of parameters can be applied for the whole data set. If it cannot, 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 an image and overlay the location of the selected stars.

```
$ ds9 gfTo_RH090523object038_chihiro.fits
```

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

Note 3: Three scripts that are used in this step, i.e., `fwhmpsf_batch.csh`, `starselect.csh` and `psfmatch_batch.csh`, may not work in crowded fields. In such fields, it may be necessary to estimate the PSF manually.

Note 4: The fwhmpsf `batch.csh` and/or `psfmatch_batch.csh` described in §4.7 may not work properly for the frames with significant amounts of cosmic ray hits. If this is the case, we suggest eliminating bad pixels hit by cosmic rays using software designated for this particular purpose, e.g., `L.A.Cosmic` (`http://www.astro.yale.edu/dokkum/lacosmic/`). We have checked that the IRAF version of `L.A.Cosmic` has successfully removed such pixels hit by cosmic rays. If you use this package, apply it to the data immediately after flat fielding (`fTo_*.fits`).

4.7 Equalize the PSF Size (Step 7)

```
$ 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` attempts to match the PSF size of all images to be combined to a predetermined target FWHM. Images with PSF sizes smaller than the target (within a small range) are Gaussian smoothed. Other images are simply copied. The target FWHM should represent the typical PSF size for the exposure, having the worst (i.e., the largest) PSF size among the exposures to be combined. In the case of the sample data, we adopt `target FWHM` = 4.1 as a fiducial value based on the results obtained in Step 6.

```
Example:
  $ ls -1 gfTo_RH090*.fits > psfmatch_batch.lis
  $ psfmatch_batch.csh psfmatch_batch.lis 50 2000 40000 2.0 7.0 4.1
```

The command produces output like:

```
pgfTo_RH090523object038_chihiro.fits 4.10 0 5 15 0 0
pgfTo_RH090523object038_clarisse.fits 4.10 0 2 16 18 0
pgfTo_RH090523object038_fio.fits 4.10 0 0 16 18 0
  ...
```

```
PSF | number of images
3.9 |***
4.0 |***
4.1 |******************************
4.2 |************
4.3 |**
```

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

1st : Name of image
2nd : Mean FWHM of PSF after PSF matching in pixel unit
3rd : Number of objects having PSF sizes within 0.1 pixels centered on (mean PSF − 0.2 pixel)
4th : Number of objects having PSF sizes within 0.1 pixels centered on (mean PSF − 0.1 pixel)
5th : Number of objects having PSF sizes within 0.1 pixels centered on mean PSF
6th : Number of objects having PSF sizes within 0.1 pixels centered on (mean PSF + 0.1 pixel)
7th : Number of objects having PSF sizes within 0.1 pixels centered on (mean PSF + 0.2 pixel)

An ASCII histogram following the log illustrates the distribution of mean PSFs.

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

4.8 Subtracting the Sky Background (Step 8)

```
$ 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` (1) computes a mesh pattern that represents the sky background, (2) interpolates the pattern, and (3) subtracts it from the image. The script creates a grid — referred to as "sky-mesh size squares" — on the image. The grid spacing having is half of the "sky-mesh" size. An appropriate sky-mesh size will be selected for each mesh, and assigned to the pixel located at the center of the mesh. After rejecting the outliers, the sky values for other pixels will be given by interpolating bi-linear from the surrounding meshes. Note that the sky-mesh size must be at least twice the size of the largest object of interest. This is due to the Nyquist sampling theorem.

```
Example:
  $ ls -1 pgfTo_RH090*.fits > skysb.lis
  $ skysb.csh skysb.lis 64
```

Once the sky background level is subtracted, the background of an image should be around zero without a spatial gradient. If there is an extended object(s) spreading over a large fraction of the image, the algorithm will most likely fail. Subtraction of sky background in crowded fields requires special data handling and you will need to estimate the sky background manually.

4.9 Masking the AG Shade (Step 9)

```
$ 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 *chihiro*, *clarisse*, *fio*, *kiki*, and *nausicaa*. Other files are not affected.

```
Example:
  $ ls -1 spgfTo_RH090*.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. To identify images where a shadow of the AG probe appears, look for images whose pixels at the top edge have the masking value of −32768 over more than a hundred pixels. Note that the shadow of the AG probe does not appear all the time.

4.10 Masking Bad Pixels (Step 10)

Data in some pixels may be corrupted due to defects of detectors and/or other problems that may have occurred during the observation. Such regions should be consistent across the exposures (i.e., they are not time variable), and should be masked accordingly. For instance, we suggest masking the background areas where flattening fails and systematically deviates from zero. If plenty of exposures cover the observed region, we suggest not spending much time with this step. This is because outliers will be rejected automatically in Step 12. The SDFRED2 package offers three methods — linear, circular, and rectangular — to specify regions to be masked for eliminating bad pixels. Here, "linear region" connects the two points (x_1, y_1) – (x_2, y_2), extends the line to the edges of the image, and masks the pixels within "width" from the line. The "circular region" masks the pixels in a circle. The "rectangular regions" masks rectangular regions aligned to the pixel coordinates.

```
$ 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
```

Linear region The script line bank masks a linear structure such as satellite trails.

```
Example:
  $ line_blank AspgfTo_RH090523object038_kiki.fits \
    10 3826 1351 8 90 -32768 lAspgfTo_RH090523object038_kiki.fits
```

The example shown creates a mask for a line which crosses (10, 3836) and (1351, 8) and around 90 pixels wide.

```
Example:
  % cat lblank.txt
  line_blank AspgfTo_RH090523object038_kiki.fits 10 3836 1351 8 90
    -32768
  lAspgfTo_RH090523object038_kiki.fits
```

The "line_blank" command (above) to `AspgfTo_RH090523object038_kiki.fits` masks a rectangular region with a width of 90 pixels, centered on $(X, Y) = (10, 3836), (1351, 8)$.

```
  % bash < lblank.txt
```

After execution, you will find the resultant masked images with the names shown below.

```
lAspgfTo_RH090523object038_kiki.fits
lAspgfTo_RH090523object038_sheeta.fits
lAspgfTo_RH090523object038_sophie.fits
```

Check whether or not you masked properly by comparing the input and output images.

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

input image = name of image to mask
blank list = a text file describing the x and y coordinates,
as well as the radii of the areas to be masked
blank value = mask value (usually -32768)
output image = name of masked image
```

Circular region The script `circular_blanks` masks circular regions.

```
Example:
  $ circular_blanks lAspgfTo_RH090523object038_chihiro.fits \
    blanklist -32768 clAspgfTo_RH090523object038_chihiro.fits
```

where `blanklist` looks like:

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

The two lines correspond to a circle of $(x, y, r) = (365, 1835, 80)$ and $(x, y, r) = (1202, 3582, 100)$. Here, we stress that the command (above): `clAspgfTo_RH090523object038_chihiro.fits` represents an example of how to use it. In practice, you probably will not need to mask a circular region (likely the case for the other frames in the training data set).

```
$ blank.csh [blank list]

blank list = list of images to be masked
```

Rectangular regions For each image, `xxx.fits`, in the blank list the script `blank.csh` will look for a file named `blankmap_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_RH090*.fits > blank.lis
  $ ls -1 lAspgfTo_RH090*.fits >> blank.lis
```

Exclude files below from blank.lis.

```
AspgfTo_RH090523object038_kiki.fits
AspgfTo_RH090523object038_sheeta.fits
AspgfTo_RH090523object038_sophie.fits
AspgfTo_RH090523object039_ponyo.fits
AspgfTo_RH090523object039_san.fits
AspgfTo_RH090523object039_satsuki.fits
AspgfTo_RH090523object039_sheeta.fits
AspgfTo_RH090523object039_sophie.fits
AspgfTo_RH090523object042_sophie.fits
```

There are `lAspgfTo_*.fits` files made by line blank for these data.

```
  $ blank.csh blank.lis
```

Mask files have been included for a subset of images

```
blankmap_lAspgfTo_RH090523object038_sheeta
blankmap_lAspgfTo_RH090523object038_sophie
```

These files have entries like:

```
$ cat blankmap_lAspgfTo_RH090523object038_sheeta
1965 2030 2376 2552
```

The script masks the regions specified in the corresponding `blankmap_xxx` file. If the `blankmap_xxx` file does not exist, the script will simply copy the image file to the output.

4.11 Estimating Alignment and Scaling (Step 11)

```
$ 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 ratio of objects per pixel
                       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 ratio (S/N) can be improved by combining multiple images (if you have them) to produce a final image. The script `makemos.csh` determines the shifts, rotations, and flux scales of different images. The script identifies stellar objects in each image and determines the shifts, rotations, and flux scale from objects common to multiple images. The first image in the list is used as the reference image.

```
Example:
  % ls -1 bAspgfTo_RH090*.fits > makemos.lis
  % ls -1 blAspgfTo_RH090*.fits >> makemos.lis

  $ makemos.csh makemos.lis 5 30 1000 40000 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 = 119
...
```

The script is likely to fail if the number of selected stars per image is either small ($<$ 30) or very large ($>$ 1000). Optimizing key parameters such as `[starsel nskysigma]`, `[starsel npix]`, `[starsel peakmin]`, and `[starsel peakmax]` will help the script to select appropriate stellar objects.

The best parameters for selecting objects in this step may be different from PSF measurement for many cases. This is because a different underlying algorithm is employed in order to find a wider range of objects to determine relative positions and flux scaling that work over a range of fluxes.

```
Example:
  $ cat all.mos
  bAspgfTo_RH090523object038_chihiro.fits 0.000000 0.000000 0.000000
    1.000000
  bAspgfTo_RH090523object038_clarisse.fits 2075.014139 1.531898
    -0.000102 1.017556
  bAspgfTo_RH090523object038_fio.fits 4184.051440 1.028001 -0.000054
    1.079106
    ...
```

As shown above, you will see five parameters (i.e., columns) in the output `*.mos` file: the name of the image, the x offset, the y offset, the counterclockwise rotation (radian), and flux ratio. If each result has four output parameters followed by the image name, the alignment or/and scaling have finished successfully. If the alignment and scaling have failed, the output file may not be produced at all, miss some parameters, or have unreasonable values.

We – the SDFRED support team – have been making continuous efforts to provide users more sophisticated method(s) that examines `all.mos`. We wish to share the following tips:

1. Inspecting the final image created in **Step 12** must be done. However, bear in mind that it is not the ultimate method. If the number of exposures is large, it is difficult to detect some small defects by visual inspection in the final image.

2. It is always a good idea to make a plot of the second vs. third columns stored in `all.mos`. The result shows the relative position of each shot, and represents the dither pattern as well as the chip positions. If there is a large leap in value, the matching has failed.

3. The distances between CCD chips should be almost constant (a slight difference may exist due to atmospheric dispersion between chips). If the distances between any arbitrarily chosen chip pairs for the same exposure (e.g., between *chihiro* and *sheeta*) has changed significantly across exposures, the data of the corresponding exposure on another chip would be incorrect.

4. The fifth column of `*.mos` (relative flux) of a chip should be almost proportional to the exposure time, if the sky condition is photometric (it is affected by atmospheric extinction (airmass), however).

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 you don't need to combine, you can skip Steps 11 and 12, and end the reduction. If you intend to combine images toward more than two fields, make sure that these data have been taken contiguously. If this is not the case, Step 11 will fail.

4.12 Combining (Step 12)

```
$ 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 output 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
```

Details and optional parameters of `imcio2a` can be printed using the command

```
$ imcio2a -h
```

Note 1: We have indicated that the image listed at the very first row of the list file (`makemos.lis`) is used as a reference for the coordinates. Even if the World Coordinate

System (WCS) of the first image is not TAN, the mosaicked output image is forced to have TAN. For instance, some images retrieved from SMOKA (http://smoka.nao.ac.jp) may have WCS of TNX. If this is the case, the WCS in the output image obtained by this is highly likely incorrect. If you wish to attain highly accurate astrometry, you must calibrate the WCS of the resulting image.

Note 2: makemos.csh assumes that the users supply images whose WCS is described by TAN and the positions given in a priori is not so different from what is expected.

Note 3: If you use -dist peak for [parameters] of the imcio2a, each pixel of the output image will have differences between the maximum and median values (i.e., maximum – median) of the corresponding pixels of the input images. This option is for checking moving objects (e.g., minor planets, and comets) a few of which are often found in one FoV of Suprime-Cam. We suggest using this option for checking the final image. If you find many bright pixels in specific regions of the final image, the alignment and scaling of input images (Step11) may fail around regions.

5 Reducing Standard Object Data

Steps 1S through 4S describe a typical procedure for reducing standard stars data. Since the physics behind this is the same as for reducing target objects, you can essentially repeat the procedure. Don't forget to work in the `standard/` directory. The flat frames must be the same as those used for the objects. Therefore be sure to copy them from the `object/` directory.

5.1 Renaming (Step 1S)

```
$ 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/
```

takes you to the directory of standard frames,

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

The `namechange.lis` should be as follows.

```
$ cat namechange.lis
SUPA01099710.fits
SUPA01099711.fits
SUPA01099712.fits
...
```

The resulting files are renamed as follows:

```
H090523object021_chihiro.fits
H090523object021_clarisse.fits
H090523object021_fio.fits
...
```

5.2 Subtraction Overscan and Bias (Step 2S)

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

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

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

```
Example:
  $ ls -1 H090*.fits > overscansub.lis
  $ overscansub.csh overscansub.lis
```

```
  $ cat overscansub.lis
  H090523object021_chihiro.fits
  H090523object021_clarisse.fits
  H090523object021_fio.fits
  ...
```

and, To_RH090523object021_chihiro.fits ... are created.

5.3 Flat Fielding (Step 3S)

```
$ 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 flat frames used in this step (ffield mf.lis) must be identical to those used for the target(s) in order to cancel out uncertainty in the normalization.

```
Example:
  $ cp ../object/dome_mflat*.fits .
  $ ls -1 dome_mflat*.fits > ffield_mf.lis
  $ ls -1 To_RH090*.fits > ffield_im.lis
  $ ffield.csh ffield_mf.lis ffield_im.lis
```

and fTo_RH090523object021_chihiro.fits ... are created.

5.4 Distortion Correction and Atmospheric Dispersion Correction (Step 4S)

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

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

The distortion correction is required since it slightly changes the sizes of the pixels, yielding slightly different flux value(s).

```
Example:
  $ ls -1 fTo_RH090*.fits > distcorr.lis
  $ distcorr.csh distcorr.lis
```

and `gfTo RH090523object021_chihiro.fits` ... are created.

5.5 Correction of Relative Flux Scale Among Chips (Step 5S)

Recall that the relative flux "`scale`" among different CCD chips has not yet been corrected even after Step 4S. For example, `*.mos` file produced at the Step 11 is as below.

```
Example:
  $ cat all.mos
  bAspgfTo_RH090523object038_chihiro.fits 0.000000 0.000000 0.000000
    1.000000
  bAspgfTo_RH090523object038_clarisse.fits 2075.014139 1.531898
    -0.000102 1.0 17556
    ...
  bAspgfTo_RH090523object038_ponyo.fits -0.083782 -4201.238704
    -0.000492 0.80 5606
    ...
```

You will probably notice that the sensitivity of *ponyo* is about 80% with respect to *chihiro*. For instance, a star that has 10,000 ADU in *chihiro* would have ∼8000 ADU if it was observed with *ponyo*. Clearly, such a relative flux scale should be corrected according to the flux ratio described in the fifth column of the `*.mos` file. Of course, this step is not required if the standard star(s) has fallen only onto *chihiro*. However, because standard stars are detected in several chips in the case of the sample data, this process is definitely required. You should divide the data by typical relative flux scale of the chip to the reference chip, which is *chihiro* for the sample data. Currently, the script for this correction is not provided. Users should do this process manually.

A Special Note for Some Suprime-Cam Data

We changed out the CCD chips in Suprime-Cam in July 2008. SDFRED2 is designed to work with data taken with the new CCDs (FDCCD). If you want to reduce data taken before June 2008, use SDFRED1. In addition, there are known problems in the data taken as of July 2008, as summarized below. The information below is essentially the same as that shown in the SMOKA web page (`http://smoka.nao.ac.jp/about/subaru.jsp`). Since the webpage may be updated more frequently than this Cookbook, we strongly suggest checking the web page. Finally, the web page detailing procedures for data taken before June 2008 (`http://smoka.nao.ac.jp/help/help supdetailNEW.jsp`) may aid you in resolving any issues.

Note 1: The data taken between July 29, 2008 and December 3, 2008,

`FRAMEID` : `SUPA01000001 - SUPA01055389`

are known to have problems in their linearity in the low counts. Their linearity may be with an accuracy of 2–5 % for < 500 ADU. Therefore, we strongly suggest not using low-count data taken during this period. Since we changed the threshold voltage for the readout system in December 2008, this problem is resolved for the data taken as of December 24, 2008 (on and after SUPA01155570).

Note 2: The data taken on September 17, 2010 in the sequence

`FRAMEID` : `SUPA01238890 - SUPA01240459`

were not properly read at the left-most channel of the CCD (`DET-ID` = 9 (*san*)). Except for this channel, we have double-checked that all the remaining data were accordingly read. Since we have optimized the readout voltage for the corresponding channel of "*san*" in October 2010, this problem is resolved for the data taken as of October 5, 2008 (on and after SUPA01240460).

Note 3: As described above, the voltage of the readout system has changed a few times. Therefore, we strongly suggest using special caution not to mix data that were taken under different voltage settings when you make a flat frame.

For all the CCDs, do not mix the following:

```
SUPA01000001 - SUPA01055389
SUPA01155570 -
```

As for a CCD of `DET-ID`=9 (*san*), do not mix the followings

```
SUPA01155570 - SUPA01238889
SUPA01238890 - SUPA01240459
SUPA01240460 -
```

Note 4: The following data are known to have errors in their FITS headers:

```
a) FRAMEID : SUPA01141740 - SUPA01141759 (two shots)
             SUPA01196480 - SUPA01196489 (one shot)
```

The position information (RA, DEC, RA2000, DEC2000, CRVAL1, CRVAL2, CRPIX1, CRPIX2) in the above data are wrong. We suspect that the other header information is also wrong. Due to these errors, SDFRED2 cannot handle these files.

```
b) FRAMEID : SUPA01239571 - SUPA01239630 (six shots)
```

The counters for these FITS files are wrong: they are shifted by one. The EXP-ID, which is taken from the last digit of the first FITS file of an exposure set, must be zero. Namely, they should be renamed as SUPA01239570 − SUPA01239579, which constitutes a set of exposure data.