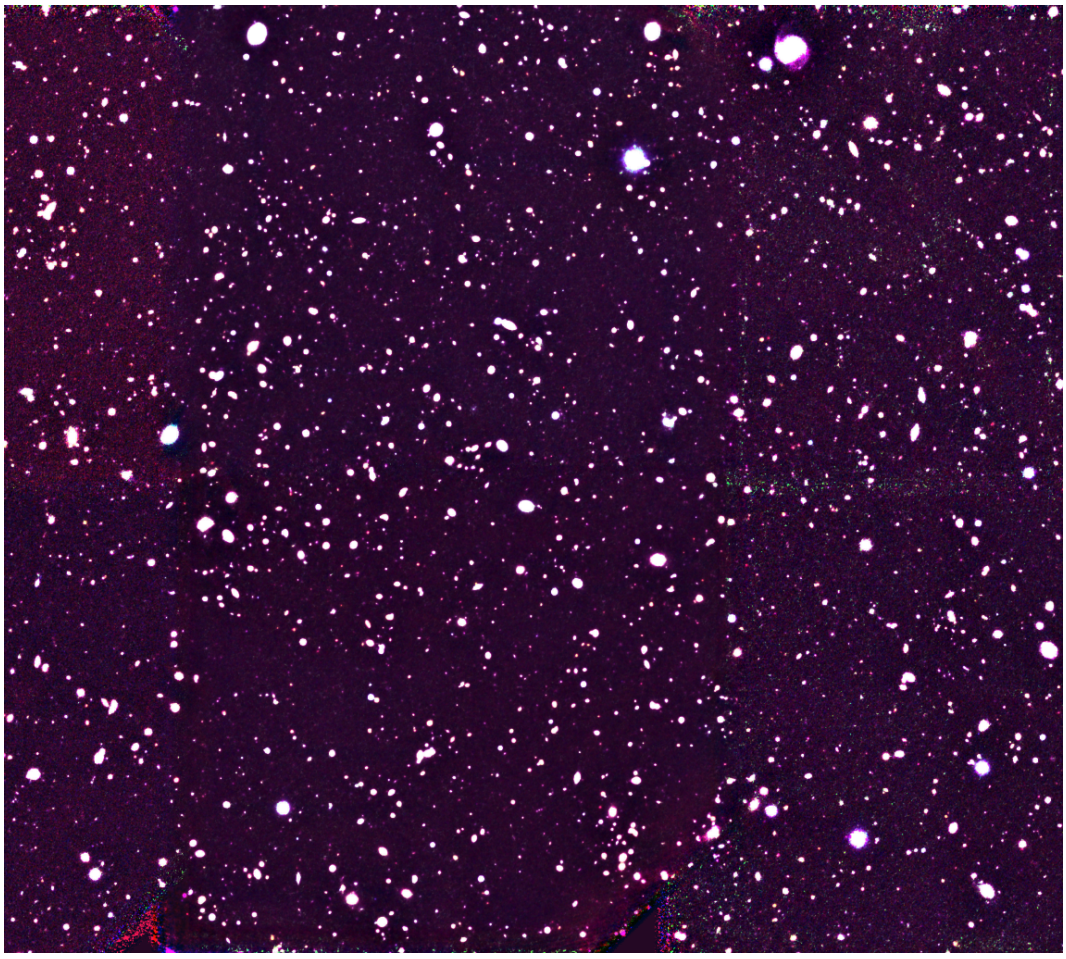
Subaru Data Reduction CookBook: MOIRCS Imaging Observations with MCSRED

— Version. 1.0.1e (May 23, 2012)—



Based on the textbook in Japanese by M. Kajisawa & I. Tanaka
for the Subaru Data Reduction School held in December 2006

Editor of English Version: R. S. Furuya,
together with the combined effort of the past and current staff at Subaru Telescope

Contents

Revision History

Version	Date	
1.0.0e	2010 January 5	First release based on the latest Japanese version (1.0j; 2009 May revised by I.T.) by R.S.F. & I.T.
1.0.1e	2010 May 6	Minor corrections by R.S.F. & A.G.

1 Foreward

This COOKBOOK teaches methods for reducing imaging data taken with MOIRCS. It was originally written in Japanese by Dr. M. Kajisawa for the Subaru Data Reduction School held in December 2006. The Japanese version has been maintained by Dr. I. Tanaka, and the latest one was used at the 2009 Subaru Spring School. Here, the authors focus on describing the MCSRED package — a series of IRAF scripts — which has been developed to reduce imaging data taken with MOIRCS. If you are new to the basics of near-infrared observations and/or data reduction, we suggest reading the general textbooks.

Once you understand how MCSRED works, we strongly recommend that you read the source codes, a.k.a., the CL scripts, that are part of the package. This is because you are most likely to get a better result by modifying these source codes and/or some parameters. These source files can be found at the `MCSRED/` directory which is under the directory where you extracted the package. Since this COOKBOOK was edited on the basis of information as of 2009 December, updates of the package should exist. Therefore, it is a good idea to check the `README_*****.txt` files (**** indicates version of release) that can be found in the tarball you downloaded. All the updates will be presented in these README files, so read these files first every time you encounter any problem.

MCSRED is a currently active project in the sense that it is always being updated based on feedback and/or requests from users. Therefore, we greatly appreciate any comments and/or suggestions you have that will improve both MCSRED and this COOKBOOK.

January 5, 2010

I. Tanaka (on behalf of the authors of the original version in Japanese)
E-mail : ichi "at" subaru.naoj.org

R. S. Furuya (current editor of English version)
E-mail : rsf "at" subaru.naoj.org

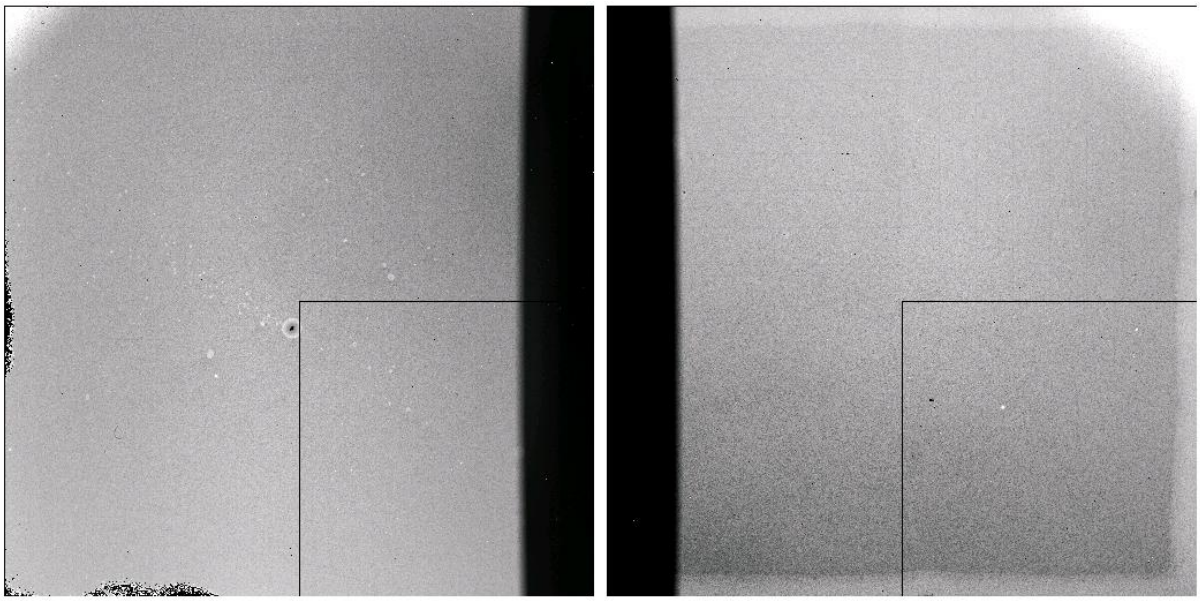


Figure 1: Raw data taken with MOIRCS; the left and right panels are those taken with channels-1, and -2, respectively.

2 Near-infrared Observations and MOIRCS

2.1 Near-infrared Observations

If you are not familiar with near-infrared (NIR) observations in general, such as observing technique and data reduction procedures, we suggest reading Chapters 2 – 5 of the "Subaru Data Reduction Cookbook: Imaging Observations with IRCS"*. You may already be aware that a typical data reduction procedure for NIR imaging is similar in principle to that for CCD imaging observations at optical wavelengths. We suggest that novice users have a look at the "Data Reduction Manual for Subaru Suprime-Cam"[1] as well.

In general, imaging observations in NIR may be different from those in the optical regime in the sense that:

- the sky background level in the NIR is not only significantly higher, but also highly time variable, and
- NIR detectors do not use CCDs as used in the the optical regime, and the sensitivity difference between pixels is larger than that of CCDs used in the optical regime.

Because of the high background level, we have to set shorter integration times per exposure than for optical observations. This produces large numbers of frames. Each one has a short integration time. With this reasoning, it would be almost impossible to identify your target sources in a single frame unless they are extremely bright compared to the sky background level, as shown in Figure 1. To have useful images for astronomy, like those presented in Figure 2, one has to "cook" such raw data by such means as subtracting sky background emission, for example.

*http://www.naoj.org/Observing/DataReduction/index.html

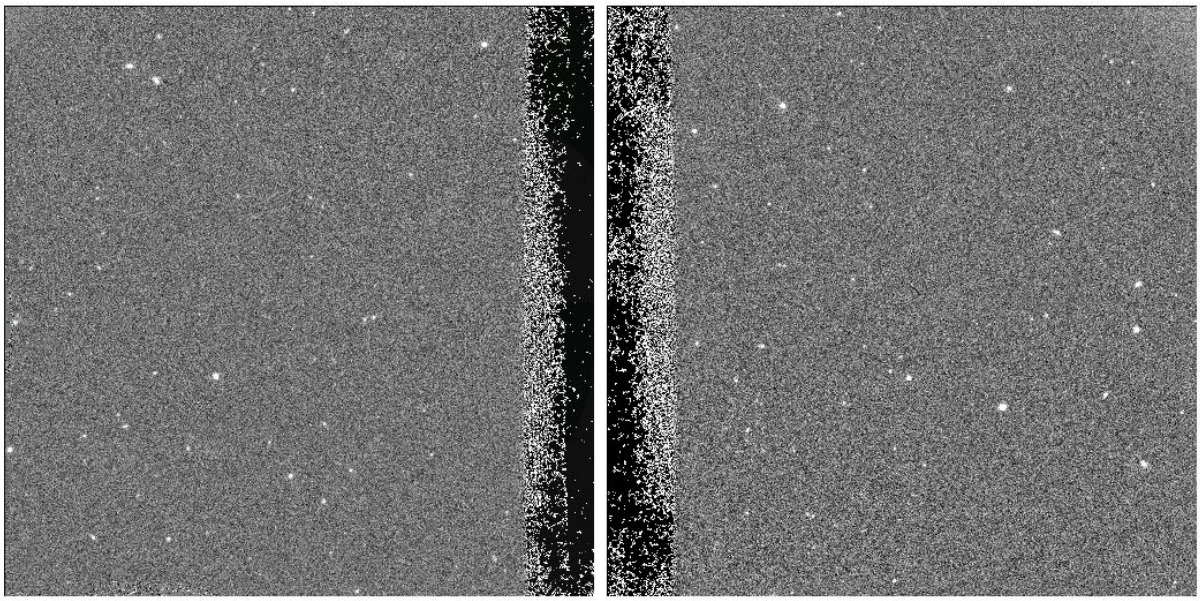


Figure 2: The MOIRCS data shown in Figure 1, after flat-fielding and subtracting the sky background emission.

An outline of data reduction process is:

1. Dark subtraction
2. Flat-fielding
3. Subtracting sky background
4. Distortion correction
5. Position shifting between individual frames
6. Adding shifted images, i.e., so-called shift-and-add.

As illustrated in Figure 3, imaging observations in optical/NIR wavelengths usually take each image with slightly shifted pointing centers. This is referred to as "dithering observations". The purpose of dithering is to avoid observing the same object(s) at the same pixel positions over many frames. This reduces statistical noise. Therefore, in order to add images taken with dithering, one has to correct for such dithering offsets so that photons from an object always fall at the same position (coordinate). This procedure is usually called "shift-and-add". The advantage of dithering observations is that one can readily make a source-emission-free image, which is supposed to represent only sky background emission, by median-averaging these dithered images. Such a median-averaged image is referred to as a median-combined sky-frame (hereafter, median-sky), and can be used for sky background subtraction. Our experience suggests that subtraction of sky background is critical to the quality of the final image. This is due to the highly time-variable nature of the infrared sky.

Subsequently, one has to make a distortion correction. This is a procedure to remove optical aberration caused by the telescope and/or the instrument. If distortion has not been properly corrected in each image, you will probably fail to shift-and-add objects existing in the outer part of field of view (FOV). In addition, if you are working on images taken of

extended emission objects'such as star forming regions, or gravitational lensing objects etc., a failure to correct distortion may cause artifacts in the morphology of these targets.

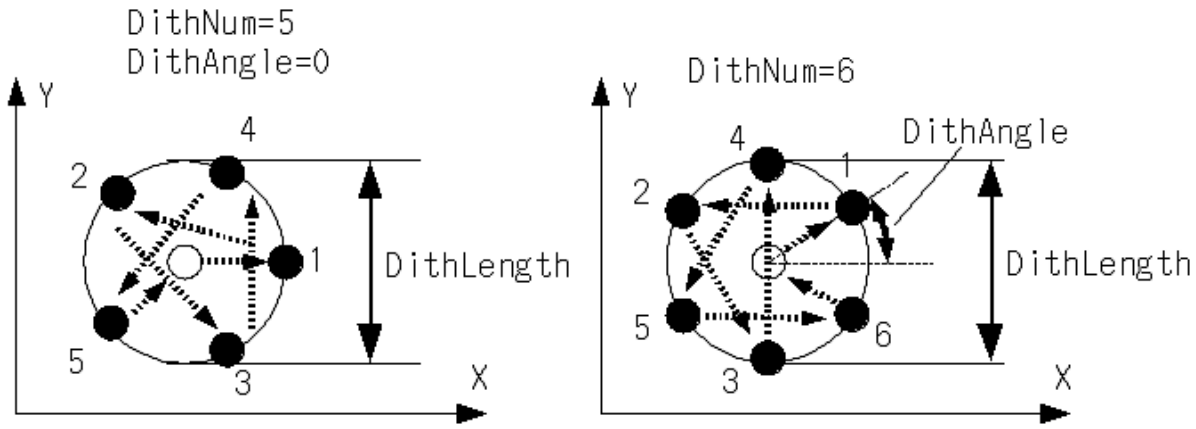


Figure 3: An example of the dithering pattern for MOIRCS observations

2.2 MOIRCS

The Multi-Object InfraRed Camera and Spectrograph, MOIRCS, is one of the instruments used at the Subaru Telescope, and has fairly high capability in both imaging and spectroscopic observations in NIR. The details, as well as information on recent updates of the instrument, can be found at `http://subarutelescope.org/Observing/Instruments/MOIRCS/index.html`. Here, we wish to give a brief introduction to how the instrument works for imaging observations.

MOIRCS has two detector arrays called HAWAII-2, which provide a wide field of view of 4×7 arcmin^2. As shown in Figure 4, we refer to the two detectors as channel-1 and channel-2. Each HAWAII-2 array has 2048×2048 pixels, with pixel scale, i.e., pixel FOV, of 0.117 arcsec pixel^{-1}. As shown in the right hand side panel of Figure 4, there is a linear gap between the two arrays which is used for spectroscopic observations.

An integration with MOIRCS produces two FITS format files from the dual detector. The FITS files have names of MCSA000?????.fits, where ????? is a five-digit number representing the frame identification (ID) number. Here, an odd frame-ID will be given for data from channel-1, i.e., Detector 1, and an even one for that from channel-2, i.e., Detector 2. Such a frame-ID for the data from Detector 2 always pairs with a frame-ID from Detector 1. For instance, MCSA00014043.fits and MCSA00014044.fits were produced at the same time for an exposure with channels 1 and 2, respectively. Notice that the x- and y-axes of these FITS data (see Figures 1 and 3) are rotated clockwise by 90 degrees with respect to the illustration in Figure 4.

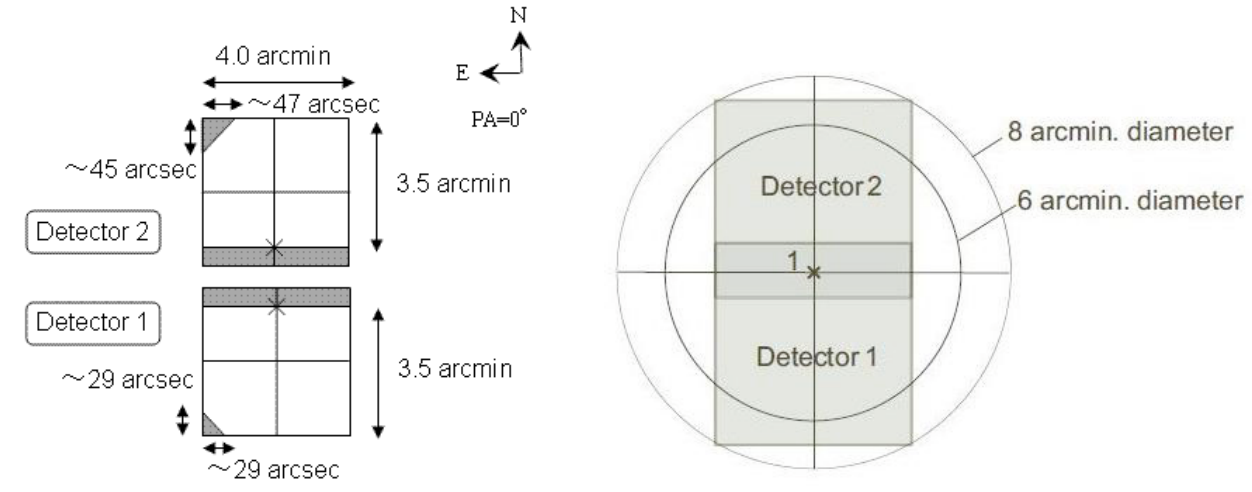


Figure 4: Field of view of MOIRCS provided by the two detectors. In the left hand figure, the shadowed regions at the bottom of Detector 2 and the top of Detector 1 are vignetted pixel regions, and are not used in the case of imaging observations. These vignetted pixel regions are used for spectroscopic observations.

3 MCSRED

MCSRED is an imaging data reduction package for MOIRCS running on IRAF[†]. The package has been developed by Dr. I. Tanaka at Subaru Telescope, National Astronomical Observatory of Japan. MCSRED consists of a series of IRAF scripts, a.k.a., CL scripts, running on version 2.12.2 or later, and the latest version has been tested on IRAF 2.14. Dr. I. Tanaka continues to develop the package to include updates of the instruments and improve the performance. We therefore recommend that readers visit the MCSRED webpage (`http://www.naoj.org/staff/ichi/MCSRED/mcsred.html`) often to check for any updates.

In this COOKBOOK, we describe how to reduce a set of imaging data using the latest version as of May 2009. The goal of this COOKBOOK is to give an overall idea what the MCSRED package offers, rather than giving complete instructions for end users.

3.1 Getting Started

The latest version of the MCSRED package can be obtained from the MOIRCS web page at `http://www.naoj.org/staff/ichi/MCSRED/mcsred.html`. Once you have downloaded the tarball and extracted it, you will find a README file (README.pdf) that should be referred to when you have any questions and/or difficulties. If you want to use MCSRED for your observations, we suggest contacting your support scientists well in advance so that they can give you instructions for MCSRED during your observation run.

[†]Image Reduction and Analysis Facility, IRAF, is a widely used data reduction package developed at National Optical Astronomy Observatory, U.S.A.

Finally, readers must keep in mind that both the distortion correction and "mosaic pattern" of the two detectors change every thermal cycle of MOIRCS. **Therefore, you MUST select an appropriate version of the configuration file for your observing period.** Details are described on page 18.

3.1.1 Installing MCSRED

As described, MCSRED consists of a series of IRAF scripts running in version 2.14. MCSRED may work on an older version of IRAF, but some task(s) may not work properly because the task(s) may be outdated or/and there were changes of some input parameter(s).

After obtaining the MCSRED tarball, uncompress it and extract the package into any directory as follows:

```
tar xvzf mcsred20090425.tar.gz
```

3.1.2 Preparation for IRAF

Since MCSRED runs on IRAF, which is also handy for general purposes, let us prepare for IRAF. To use IRAF, issue the `mkiraf` command in any directory where you want to work,

```
$ mkiraf
        -- creating a new uparm directory
        Terminal types: xgterm,xterm,gterm,vt640,vt100,etc.
        Enter terminal type:
```

Select `xgterm` here,

```
        Enter terminal type: xgterm
        A new LOGIN.CL file has been created in the current directory.
        You may wish to review and edit this file to change the defaults.
```

The above should have created the ASCII text file `login.cl`, which describes various parameters to define your IRAF environment. Although the default setting should suffice for standard users, we suggest some changes.

Add the following two sentences at any place in the ending portion of the file. However, these two lines must appear before the line of "`keep`" at the very end.

```
task $mcsred = (full path to directory you extracted tar file)/MCSRED/mcsred.cl
set dir_mcsred  = "(full path to directory you extracted tar file)/MCSRED/"
```

Modify the following sentence below the line "`# Uncomment and edit to change the defaults.`". In addition, do not forget to delete '#' mark at the head of the modified line:

#set stdimage = imt800 —> set stdimage = imt2048

To start IRAF, make sure that you are at the directory where you created the `login.cl` file, launch an xgterm by typing `xgterm &`, and type `cl` on the terminal.

$ cl

```
NOAO PC-IRAF Revision 2.12.2-EXPORT Sun Jan 25 16:09:03 MST 2004
This is the EXPORT version of PC-IRAF V2.12 supporting most PC systems.

Welcome to IRAF.  To list the available commands, type ? or ??.  To get
detailed information about a command, type 'help command'.  To run a
command or load a package, type its name.  Type 'bye' to exit a
package, or 'logout' to get out of the CL.  Type 'news' to find out
what is new in the version of the system you are using.  The following
commands or packages are currently defined:

  apropos     images.     mscred.     plot.       stsdas.     utilities.
  dataio.     language.   noao.       proto.      system.
  dbms.       lists.      obsolete.   softools.   tables.

cl>
```

If you don't get the series of messages above, you failed to start IRAF. Most likely, you may have started IRAF in wrong directory. Some tasks in MCSRED assume that users do not change the default settings of input parameters for the implemented IRAF tasks. As we have just started IRAF with the newly made `login.cl`, all the IRAF parameters should have the default values. If this is not the case, or you want to clear all the input parameters for some reason, type

```
cl> unlearn iraf
```

This forces all the input parameters of your IRAF environment to the the default values. If you want to save the current parameter settings before resetting them, save a backup copy of the whole `uparm/` directory somewhere. Here, the directory can be found at your IRAF "login" directory where your `login.cl` file exists.

To quit IRAF, simply type `logout`.

```
cl> logout
```

3.1.3 If You are Using IRAF v2.12.1

If you are using an older version of IRAF v2.12.1., you need to edit the IRAF script which constitutes the MCSRED package. All the source files of MCSRED package can be found in the directory where you extracted the tar file, i.e.,

```
$ cd (the directory where you extracted the tar file)/MCSRED/
$ ls
```

You should find many CL scripts ending with an extension of `.cl`. Find the lines that call the IRAF `"strldx"` in the `basename.cl` script; the IRAF task is not implemented in IRAF v.2.12.1. Move `basename_v2.12.1.cl` under the /MCSRED/CONTRIB/ directory to that of `basename.cl` by overwriting it.

In addition, readers using IRAF v.2.12.1 or earlier must edit `invmask.cl` script, which will be used in §4.2; you have to delete the `direction` parameter in the task `geoxytran` called in the `invmask.cl` script.

3.1.4 Setting Your Shell Environments

In order to use MCSRED, users must configure the shell environments accordingly, as shown next.

For bash users, add the following sentence in either `.bash_profile` or `.bashrc` which can be found under your home directory:
`export MCSRED_DIR="(full path to the extracted directory)/MCSRED"`

For tcsh users, add the below in your `.cshrc` file:
`setenv MCSRED_DIR (full path to the extracted directory)/MCSRED`

3.1.5 Installing SExtractor

To use of all the capabilities provided by MCSRED, we strongly suggest having the `SExtractor` package, which is a widely used software that identifies stellar objects and performs photometry. If your system does not have the package, get it from
`http://terapix.iap.fr/rubrique.php?id_rubrique=91` and install it using the included instructions.

3.1.6 Exploring MCSRED

After sourcing `.bashrc` (or the shell environment file you edited in §3.1.4), start IRAF to load MCSRED from unix command line, e.g.,

```
source ~/.cshrc
```

Make sure that you are at the directory where `login.cl` exists, and type,

```
$ cl
```

you will get a prompt of

```
cl>
```

Here, `cl>` is the IRAF prompt. Now, we can load the MCSRED package by typing `mcsred` at the launched xgterm.

```
cl> mcsred

basename       fringesub      mcs_mksflat    moscorcalc     randaperr
checkfrsb      gmkgtrimages   mcsall         noddata        sbselfsky
checkwdither   gsextcat       mcsdisplay     plsatellite    tiltskycor
chkgmpdata     imcheck        mcsgeocorr     prmask         tsubanomaly
cleanall       imreflection   mcsimstat      psfestimate
cutpr          invmask        mkcorrawdata   qchkfrsb
dmosimg        kage           mkdistmask     qmsepskysb
findms         listprep       mkdome         quadcor

mc>
```

Once MCSRED has been successfully loaded to your IRAF environment, you will see a prompt of `mc>` (or `mcsred>` for IRAF 2.14 or later). If not, check the `login.cl` file which defines your environments.

It is also good idea to start a FITS viewer such as `ds9` by typing,

```
mc> !ds9 &
```

Now, we are ready to reduce imaging data taken with MOIRCS.

3.2 The All-in-One Task: `mcsall`

3.2.1 Overview of `mcsall`

The `mcsall` task is an all-in-one procedure to reduce MOIRCS data automatically. In this sense, it may be thought of as a pipeline, and is designed to deal with so-called "deep field" imaging data. Given this property, `mcsall` may fail to deal with extended object(s) that fill a significant portion of the FOV. Such a failure would happen during the sky-subtraction process. Here, the "significant portion" is in terms of dithering width. For instance, if your source size is more than $\sim 1/4$ with respect to the FOV, `mcsall` would fail. If you want to observe such an extended source(s), consult with your support scientist(s); they may suggest using "nodding" observations. If you intend to reduce such a data set acquired with the `SKYNOD`, you should consider using the MCSRED package, `noddata`, designed for this purpose (see §4.1 for details).

The `mcsall` task assumes that users will supply multiple images taken by dithering (recall Figure 3). Therefore, if you repeatedly observed a single sky position without dithering, you have to mimic the pipeline by editing an input file list so that any images taken at the same position will not be listed continuously.

In addition, if atmospheric opacity and/or seeing size changed significantly during observations, you have to eliminate those frames because they will degrade the quality of the final image or/and increase image noise levels. Special caution should be used when you are working on data retrieved from the archive system.

The `mcsall` pipeline sequentially performs each data reduction step as shown below with a task whose name is shown in parenthesis :

1. Making a mask for objects (`mcs_mksflat`);
2. Flat-fielding (`imarith`);
3. Median-sky-subtraction (`sbselfsky`);
4. Subtracting residual pattern noise with a low-order function (`qmsepskysb`);
5. Correction for gaps between quadrants (`quadcor`);
6. Distortion correction (`mcsgeocorr`);
7. Identification of stellar objects in each frame (`gsextcat`);
8. Shifting-and-adding the images (`gmkgtrimages`).

In addition to the tasks above, MCSRED offers other tasks to produce a list of input FITS files (`listprep` described in §3.2.3), and to "mosaic" data taken with the two detectors (`dmosimg` described in §3.3.1). Note that these two tasks are not implemented in `mcsall`. We describe a standard `mcsall` usage in this subsection. The details of each task implemented in `mcsall` are described in Appendix A.

3.2.2 Associating FITS Files Numbers with their Observing Purposes

To demonstrate how to reduce MOIRCS data using MCSRED, we use an example data set that was taken towards a blank field of the Hubble Deep Field North (HDF-N). Here, a blank field does not contain any special objects, but has "general properties" of the universe.The HDF-N does not contain any bright (galactic) objects. Recall that `mcsall` is optimized to reduce such "deep space" images. The data here can be retrieved from the public data archive system of the Subaru Telescope, SMOKA[‡], because more than 18 months have passed since the data were taken by the original observer (in this case, December 9, 2005). Assume that we have downloaded and stored all the FITS data under the following directory

```
$ ls /home/analysis/FITS/
```

This command shows a list of FITS files whose names are MCSA000?????.fits. All the FITS file names of MOIRCS data begin with the string of `MCSA`. Notice that, in general, a set of data includes not only those of your science target, but also those for calibration purposes, such as standard stars data.

Next, we have to associate FITS file names with their purpose(s). Such information is written in the header portion of each FITS file. IRAF provides a few tasks to extract any information that users want to extract.

Below extracts all the header information of the FITS files specified here, i.e., MCSA00014027.fits.

```
mc> imheader /home/analysis/FITS/MCSA00014027.fits l+
```

You may not need to have all the output information, and you may want to select only some of it. The IRAF task `hselect` helps you for such a purpose, as follows.

```
mc> hselect /home/analysis/FITS/*.fits FRAMEID,OBJECT,FILTER01,EXPTIME
yes


MCSA00013675    DARK    J       13.000
MCSA00013676    DARK    K_CONT  13.000
MCSA00013685    DARK    J       13.000
MCSA00013686    DARK    K_CONT  13.000
MCSA00013687    DARK    J       13.000
MCSA00013688    DARK    K_CONT  13.000
MCSA00013689    DARK    J       13.000
MCSA00013690    DARK    K_CONT  13.000
 .....
```

This task, `hselect`, extracts parameter values of "FRAMEID", "OBJECT", "FILTER01", and "EXPTIME" from FITS file headers, given as the input parameter of the task. Here, "FRAMEID" is the identification number of the data frame, and the subsequent keywords

[‡]http://smoka.nao.ac.jp/index.jsp

indicate name of the observed source ("OBJECT"), the filter used for the observation ("FILTER01"), and exposure time ("EXPTIME").

Recall that the odd and even frame numbers indicate the data from channel-1 and channel-2, respectively (§2.2). The above shows that the channel-1 and -2 images were taken with the J-band and K_CONT filters, respectively, over a 13-second integration. Given its purpose, one can take dark frame(s) with any filter(s). Although not shown in the above example, scientific data are generally taken with the same filter between the dual channels.

As shown above, one can extract any desired information with the task `hselect`. If you want to save the extracted information, you can write it to an ASCII text file. Make sure that you are currently at the directory where you want to work, then issue the following command

```
mc> hselect /home/analysis/FITS/*.fits FRAMEID,OBJECT,FILTER01,EXPTIME
yes > header.txt
```

The result was written in the file you specified, i.e., `header.txt`. Check the result by, e.g.,

```
mc> less header.txt
```

You will see that the output is stored in the file; hit the SPACE key to scroll down, and find the string of HDFN as,

```
MCSA00014016    LOCKMANHOLE     KS      100.000
MCSA00014017    HDFN1   KS      100.000
MCSA00014018    HDFN1   KS      100.000
MCSA00014019    HDFN1   KS      100.000
MCSA00014020    HDFN1   KS      100.000
MCSA00014021    HDFN1   KS      100.000
MCSA00014022    HDFN1   KS      100.000
MCSA00014023    HDFN1   KS      100.000
MCSA00014024    HDFN1   KS      100.000
MCSA00014025    HDFN1   KS      100.000
MCSA00014026    HDFN1   KS      100.000
MCSA00014027    HDFN1   KS      100.000
MCSA00014028    HDFN1   KS      100.000
MCSA00014029    HDFN1   KS      100.000
MCSA00014030    HDFN1   KS      100.000
MCSA00014031    HDFN1   KS      100.000
MCSA00014032    HDFN1   KS      100.000
MCSA00014033    HDFN1   KS      100.000
MCSA00014034    HDFN1   KS      100.000
MCSA00014035    HDFN2   KS      100.000
MCSA00014036    HDFN2   KS      100.000
MCSA00014037    HDFN2   KS      100.000
MCSA00014038    HDFN2   KS      100.000
```

.....

Here, HDFN1 and HDFN2, respectively, show that the data are taken towards fields 1 and 2 (which are set by the observer) in the Hubble Deep Field North (HDF-N). The frame numbers are assigned with sequence of the observations, i.e., the above is telling us that the observations were done by repeating the sequence below.

Observing HDFN1 nine times with a 100-second integration $\longrightarrow$ Moving to HDFN2, and observing the field nine times $\longrightarrow$ Returning to the HDFN1, and observing the field nine times $\longrightarrow$ Going back to HDFN2 $\longrightarrow$ HDFN1

Check that the data were taken 9×3 times towards HDFN1, and 9×2 times towards HDFN2. You may have realized that all the data have been taken in the K_s band.

3.2.3 Preparing for a List of Input File — `listprep`

Since `mcsall` independently deals with the data from the two detector arrays, we have to prepare for list files that store names of FITS files for each array. Such list files can be created with the task `listprep`. For this purpose, one has to supply path(s) of the directory(ies) where the FITS data exist, the identification number of the first FITS file (i.e., the lowest number) and that of the last. If you want to reduce the first nine frames towards the field No.1 (HDFN1), issue the below, for example,

```
mc> listprep /home/analysis/FITS hdfn1 14017 14034
```

You will obtain two list files:

hdfn1_1.lst — the FITS file list for channel-1
hdfn1_2.lst — the same, but for channel-2

We suggest checking that the `hdfn1_1.lst` and `hdfn1_2.lst` files have odd and even number FITS files only, respectively. If not, make sure of your input parameters by checking the output messages for the target name(s), exposure time, and filter information.

3.2.4 Inspecting Images

Before running `mcsall`, we recommend eliminating the data taken during bad weather conditions. A hint for this may be obtained from the seeing size for each frame. Another practical reason is that a crude value of seeing size must be given to run `mcsall`. In this subsection, we describe how to estimate seeing sizes for individual frames.

First, we have to estimate the size(s) of the point source(s) in the frames. An indicator of the atmospheric degradation is referred to as Point-spread function (PSF), and is approximated with either a Gaussian or Moffat function whose full-width at half-maximum (FWHM) is used to describe the size of the PSF. In practice, measurement of the PSF is more or less equivalent to that of the stellar FWHM.

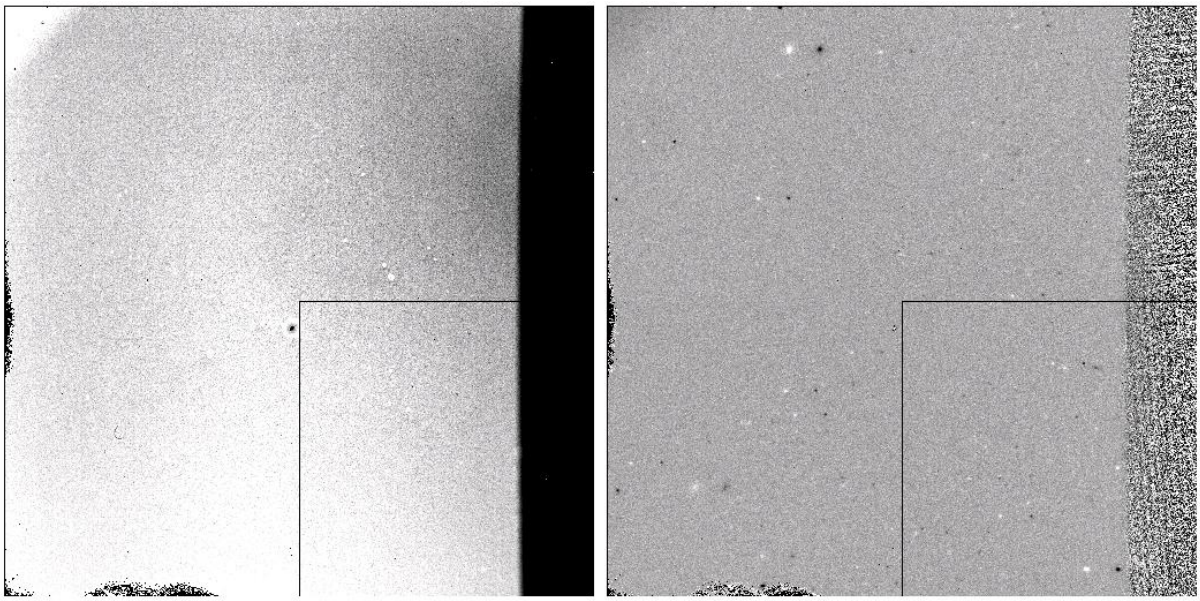


Figure 5: The left hand panel shows an example of raw data (MCSA00014017.fits). The right hand panel is obtained by dividing the raw data shown at left by its subsequent frame of MCSA00014019.fits.

Below, we describe how to measure the PSF. If you were present at the base or summit facilities during the observations, you may have rough estimates of the PSF for each exposure in your observing log. If you intend to reduce data retrieved from the archive, we suggest starting your data reduction step by inspecting the data quality.

As described in §2, it is almost impossible to recognize most of the objects in each raw data frame (see Figure 5) due to the high background noise. Dividing a frame by the one which was taken immediately after the frame helps you to assess quality of the image(s).

To display the raw data, issue the `display` command in IRAF as:

```
mc> display MCSA00014017.fits 1 zr- zs- z1=7000 z2=18000
```

Below are commands to divide the first image by the next one, and displaying the result:

```
mc> imarith MCSA00014017.fits / MCSA00014019.fits ql14017.fits
mc> display ql14017.fits 1 zr- zs- z1=0.95 z2=1.05
```

If you can recognize any point-source object(s), measure the size by typing:

```
mc> imexamine
```

Then, the cursor should appear on the image viewer (e.g., `ds9`). Move the cursor onto the object whose size you want to measure and type `a`:

```
mc> imexamine
```

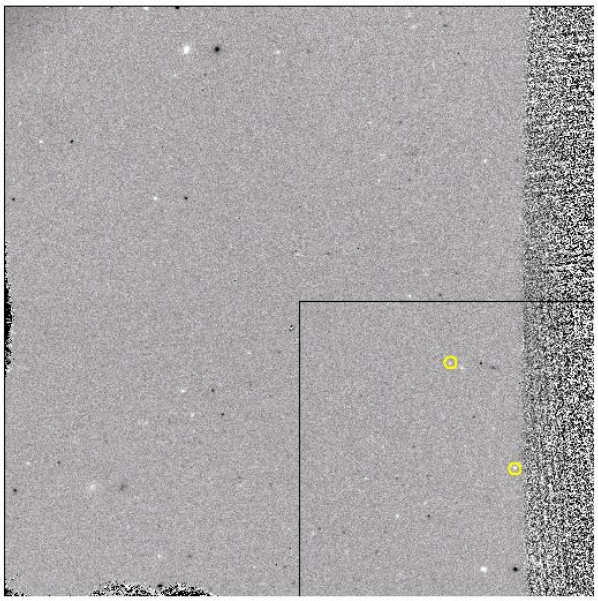


Figure 6: An example of a non-calibrated image obtained from dividing by the next frame. The two yellow circles indicate the positions of the stars.

```
#   COL    LINE    COORDINATES
#     R     MAG     FLUX     SKY     PEAK     E    PA BETA ENCLOSED   MOFFAT DIRECT
1546.00  808.41 1546.00 808.41
  17.57  31.75   2.702   1.019 0.07123 INDEF    57 3.09     6.27     5.69   5.92
```

Here, the result of ENCLOSED, MOFFAT, and DIRECT indicate the measured FWHM of the object in the pixel number unit; these three results are obtained by using different methods. You should measure FWHMs of the other objects.

```
mc> imexamine
#   COL    LINE    COORDINATES
#     R     MAG     FLUX     SKY     PEAK     E    PA BETA ENCLOSED   MOFFAT DIRECT
1546.00  808.41 1546.00 808.41
  17.57  31.75   2.702   1.019 0.07123 INDEF    57 3.09     6.27     5.69   5.92
1364.85  280.42 1364.85 280.42
  20.71  31.40   3.714   1.018   0.046 0.41    -4 1.44     7.89     7.09   6.95
1771.63  446.80 1771.63 446.80
  18.78  30.61   7.669   1.017  0.1179 0.22    88 4.10     6.49     6.33   6.25
 524.76 1381.21 524.76 1381.21
  21.89  30.88   6.022   1.018 0.06351 0.21    28 2.32     7.32     7.84   7.30
 126.43 1578.40 126.43 1578.40
  18.10  30.80   6.451   1.017  0.1055 0.15   -22 4.23     6.28     6.47   6.06
 631.54 1897.00 631.54 1897.00
  28.80  29.17   29.04   1.018  0.1134 0.06    70 3.00    13.33     9.61   9.61
```

In the results above, any point-like objects are considered to be star(s) or stellar object(s).

Figure 6 shows an image obtained by dividing it by the subsequent one. In the image, the two yellow circles indicate the positions of the stars whose FWHM are considered within the

PSF within a first order of approximation. After finishing all of your work, quit the task `imexamine` by pressing `q` on the image viewer.

3.2.5 Image Inspection Task — `imcheck`

Inspecting the seeing size of each frame, as well as assessing image quality by eye, are crucial to obtain the best-reduced image. For this purpose, MCSRED offers an image-inspection, task `imcheck`, which automatically performs image-division (§3.2.4) and `imexamine` does the same to a series of frames given in the list that the task `listprep` created (§3.2.3).

mc> imcheck hdfn1_1.lst

Check the quality of the subtracted-images displayed in the viewer and if it is satisfactory, go to the next one by quitting the implemented IRAF task of `imexam` by typing `q`.

3.2.6 "Good" Count and COADD

Next, one should check whether or not the scientific frames have "good" count values. In the case of MOIRCS, the so-called saturation level of the detectors cannot be used as the primary indicator for assessing validity of the data. This is because the saturation level of MOIRCS tends to vary with the intensity of the incident light and/or pixel locations where the photons fall. However, our experience suggests that the count levels should be around 22000 and 25000 ADU for the channel 1 and channel 2, respectively, when COADD =1. If you observed with COADD=3, we suggest the threshold level should be $22000 \times 3 \approx 66000$ ADU. Here, COADDing is a technique to integrate the signal of the received photons in the hardware memory of the detector without generating a FITS file over several exposures. This is a particularly powerful technique, especially for NIR observations where one cannot perform long integrations because of high background emission and the highly time-variable nature of the background emission. In practice, COADDing has a great advantage in that one can significantly reduce net observing time (i.e., reduce the overhead time) by reducing the time needed for generating many FITS files. Numbers for COADDing can be seen by typing:

mc> hselect @hdfn1_1.lst FRAMEID,COADD,EXP1TIME,EXPTIME yes

```
MCSA00014017    2       50.000  100.000
MCSA00014019    2       50.000  100.000
MCSA00014021    2       50.000  100.000
MCSA00014023    2       50.000  100.000
MCSA00014025    2       50.000  100.000
MCSA00014027    2       50.000  100.000
MCSA00014029    2       50.000  100.000
MCSA00014031    2       50.000  100.000
MCSA00014033    2       50.000  100.000
```

In this case, the data were acquired with COADD =2 for obtaining a net integration time of 100 seconds by COADDing two 50-second integrations. Notice that the integration time (the parameter of `EXPTIME`) we checked in §3.2.4 is given by `COADD` × `EXP1TIME`, where `EXP1TIME` is the integration time for each exposure.

3.2.7 Executing `mcsall`

If you are satisfied with all the inspection results so far, you can now run `mcsall`. As described in §3.2.1, `mcsall` processes data from the dual detector arrays separately. Since `mcsall` is a pipeline to deal with whole the reduction process, you have to supply many parameters at this stage.

In order to mosaic the fully reduced images from the dual detectors, one has to combine distortion-corrected images using the task `dmosimg` (described in §3.3.1). Here, distortion correction corresponds to Step 6 in `mcsall` (see page 10). Before running `mcsall`, one has to carefully check the input parameters shown below.

```
mc> lpar mcsall
       inlist = "hdfn1_1.lst"   The list of raw images (listprep output)
       resimg = "hdfn1_no1_ch1.fits" The name of the final resulting image
        (jump = 1)              Jump to the i-th procss
         (bye = 8)              Stop after the i-th procss
        (disp = yes)            Display the process?
      (config = "dir_mcsred$DATABASE/ana_dec05.cfg") the name of mcsred config
        (dosf = yes)            Step1: Use a self-flat instead of external flat
       (sflat = "")             Step1: the name of output sky flatframe.
     (extflat = "")             Step1: if dosf-, supply the name of flatframe.
        (dodk = no)             Step1: subtract the dark frame ?
        (dark = "")             Step1: the name of the user-supplied Dark Frame
        (rail = no)             Step1: do the rail remain on chip 1?
     (sthresh = 1.25)           Step1: threshold for object mask.
     (msksize = 4.)             Step1: object expansion size (pix).
       (mskcr = yes)            Step1: cosmic-ray cleaning during making mask?
       (nself = 3)              Step3: 2 x nself frames are used for making sky
     (skipqms = no)             Step4: skip qmsepsky process (cautious!)?
     (moresky = no)             Step4: more constant sky subtraction after qmse
       (crrej = yes)            Step5: cosmic-ray cleaning during mcsgeocorr?
       (minfw = 5)              Step7: Minimum FWHM for matching catalog
       (maxfw = 15)             Step7: Maximun FWHM for matching catalog
       (satur = 22000)          Step7: Saturation counts
       (thres = 5)              Step7: SExtractor: detect_thresh
        (conn = 16)             Step7: SExtractor: connected pixel
     (combine = "average")      Step8: type of final combine?
      (reject = "sigclip")      Step8: type of rejection
      (weight = "sigma")        Step8: weight -sigma or exptime?
      (lsigma = 3)              Step8: lower sigma clipping factor
      (hsigma = 3)              Step8: upper sigma clipping factor
        (nmat = 40)             Step8: nmatch parameter in xyxymatch
        (mode = "q")
```

Here are some hints for selecting the most suitable parameters for your data:

`inlist` — The name of input FITS file list created with task `listprep`.

`resimg` — Name of the final output image; an extension of `.fits` with lower case must be given.

`jump` — If `mcsall` fails at some step in the pipeline (see page 10), a parameter in `jump` allows you to restart the pipeline from your desired step.

`bye` — If you want to force `mcsall` to quit after completing a certain step, specify the task name with a parameter of `bye`.

`disp` — This parameter tells the pipeline whether or not you want to display intermediate results while the pipeline processes. In general, we strongly recommend that you inspect such intermediate products at each step. Before running `mcsall`, make sure that you have an active FITS viewer. However, if there is some reason why you cannot launch an image viewer, e.g., reducing data remotely through a network, you may want to turn off the display function by specifying this parameter.

`config` — Specifying a configuration file for MOIRCS distortion correction. **Keep in mind that users have to select an appropriate configuration file for the observing period because distortion correction parameters vary after completing every thermal cycle of MOIRCS.** Select a configuration file for your data from the list below, where we show the name of the files and the observing periods.

ana_jan05.cfg	2005 January (Test observations)
ana_dec05.cfg	2005 December (Test observations)
ana_jan06.cfg	2006 January – March
ana_mar06.cfg	2006 April – July
ana_aug06.cfg	2006 August – October
ana_nov06.cfg	2006 November – 2007 January
ana_feb07.cfg	2007 February – March
ana_may07.cfg	2007 April – June
ana_aug07.cfg	2007 July – early October
ana_oct07.cfg	2007 late October – 2008 February
ana_feb08.cfg	2008 February – July
ana_aug08.cfg	2008 August – 2009 July

In the case of the sample data, since they were taken on the night of December 9, 2005, select `ana_dec05.cfg`. The latest information about configuration files is contained in the README file of the MCSRED package.

`dosf, sflat, extflat` — `dosf` specifies whether or not to use self-flat frame. If you intend to use self-flat, change this parameter to "yes", and give the file name of the self-flat

frame to be used. If you don't use self-flat and instead use another type of flat frame (e.g., dome-flat), change `dosf=no`, and give the file name of the flat to be used in the `extflat` parameter. In MOIRCS observations, it is known that usage of "dome flat" can reduce unwanted object effects, i.e., artifacts generated by detected objects and/or that of fringe (see §4.2 and §4.3). However, "dome flat" has a disadvantage in that the pattern of sensitivity over the pixel arrays differs from that obtained from "self-flat" because the light paths for dome-flat and that for self-flat are not identical. If you decide to use dome-flat, you have to create it from the frames that have been obtained during the same observing period. This is because pixel-to-pixel sensitivity is known to change around 2–3% over thermal cycles. In the case of the sample data, we use self-flat; set `dosf` as "yes", and give a name of the flat frame in the `sflat` parameter.

`dodk, dark` — `dodk` tells the pipeline whether or not you want to subtract the dark frame. If you do, then supply a name of the dark-frame with the `dark` parameter. **It should be noted that we generally do not subtract the dark frame for MOIRCS data** because its contribution is small enough to ignore. In addition, in the case of MOIRCS, there is another source (or sources) that increases count values more than dark does. Here, these contribution are from latent, i.e., persistence, i.e., persistence, that varies with the incident angle of incident photons to the detectors. The contribution from persistence which is estimated to be approximately 2–3%, or no more than 8% of the input sky count, may cause systematic error in the making of flat frame.

`rail` — It sometimes happens that about 100 pixels from the left hand side boundary of channel-1 data are left blank because the safety rail for the MOS mask had mistakenly been left on after MOS observations. If you find such a blank region, set `rail` = yes.

`sthresh, msksize` — `sthresh` is the detection threshold in σ when identifying objects at Step 1, as described on page 10. Another parameter of `msksize` controls how many adjacent pixels users want to add to define the area size of objects. For many cases, the users do not need to change these values from the defaults unless they have strong reasons to do so.

`mskcr` – This parameter tells the pipeline whether or not to remove pixels hit by cosmic rays, using the IRAF task `cosmicrays`. If you performed a long integration with a narrow-band filter, we suggest setting `mskcr` of "yes".

`nself` — This is a parameter used to specify how many frames to be used for making a median-sky-frame, a.k.a.,"median-sky". A median-sky for a particular frame will be created from a total of 2×`nself` frames that have been obtained immediately before and after the particular frame (`nself` frames for each). For many cases, `nself`=3∼4 may be selected, which should be optimized by considering the integration time of each frame and/or sky conditions during the observations (see §A.2).

`skipqms` — If you want to remove the residual of the sky subtraction routine by `qmsepskysub`, set `skipqms` = yes. The default is `no`.

`moresky` — If you want to perform a further subtraction of the residual sky emission by means of median subtraction, set this parameter to "`yes`". In many cases, you can skip

this process.

`crrej` — This parameter tells the pipeline whether or not to correct cosmic ray signals. Since distortion correction tries to interpolate information about the strength of the signal measured at each pixel, if you do not correct pixels hit by cosmic rays and/or hot-pixels, artifacts will appear around these pixels. Therefore, the correction should be applied before making distortion corrections (the default is `yes`).

`minfw, maxfw` — These two parameters are used to set the minimum (`minfw`) and maximum (`maxfw`) FWHMs of the objects that will be used for shift-and-add images. We suggest giving a `minfw` value slightly smaller than the PSF, which helps avoid bad-pixels, whose sizes are significantly smaller than stellar sizes. This is because such very small bad pixels tend to affect the shift-and-add process. On the other hand, `maxfw` may be set to be $\sim 2-3$ times the FWHM. If you allow for more extended object(s), the final accuracy of image registration may degrade in general.

`satur, thresh, conn` — These three parameters, used when `mcsall` calls the `SExtrator` package, set detection thresholds for objects used in the image-registration process; these parameters correspond to the parameters of SATUR_LEVEL, DETECT_THRESH, DETECT_MINAREA in SExtractor, in this order. All the data above them will not be used. Notice that we have to have point-like sources to attain sensible accuracy in the image-registration process, hence, if we use any saturated object(s), the resulting accuracy is highly likely to be degraded. The `conn` parameter may be set to be $\sim 2-3$ times that of FWHM. As for `thresh`, we suggest giving a value larger than that usually used for the `DETECT_THRESH` in SExtractor (see §A.6 for the details). However, if the number of the detected object(s) is not enough, the pipeline may fail. We therefore suggest trying with `satur` of 2 to 5. In order to avoid saturation of the detectors, one must keep the default count levels at ∼22000–25000 ADU, as described in §3.2.6. Since the pipeline automatically corrects for the saturation threshold for the data taken with COADD, users do not need to consider such a correction.

`combine, reject, weight, lsigma, hsigma` — These parameters are used to make the final image by combining all the fully reduced images. If you intend to combine an image with five or six frames (or less), we strongly suggest setting `combine=average`. If a large number of images are available, a choice of `combine=average` would result in a better S/N-ratio by optimizing "reject" parameters, i.e., `pclip, sigclip`, and `avsigclip`. Here, `weight` may be selected by either estimating the RMS of the images or giving a weight on the basis of exposure times (which can be found in the header of each image). In general, the sky background level at NIR is highly time-variable, so we suggest adopting the former option.

`nmat` — This is the maximum number of the objects that will be used in the "catalog matching" process; the pipeline uses the IRAF task of `xyxymatch` for detecting objects and calculating their positional offsets. For many cases, users can keep the default value. We strongly discourage increasing `nmat` of a value more than 50, as it eats a lot of computer resources.

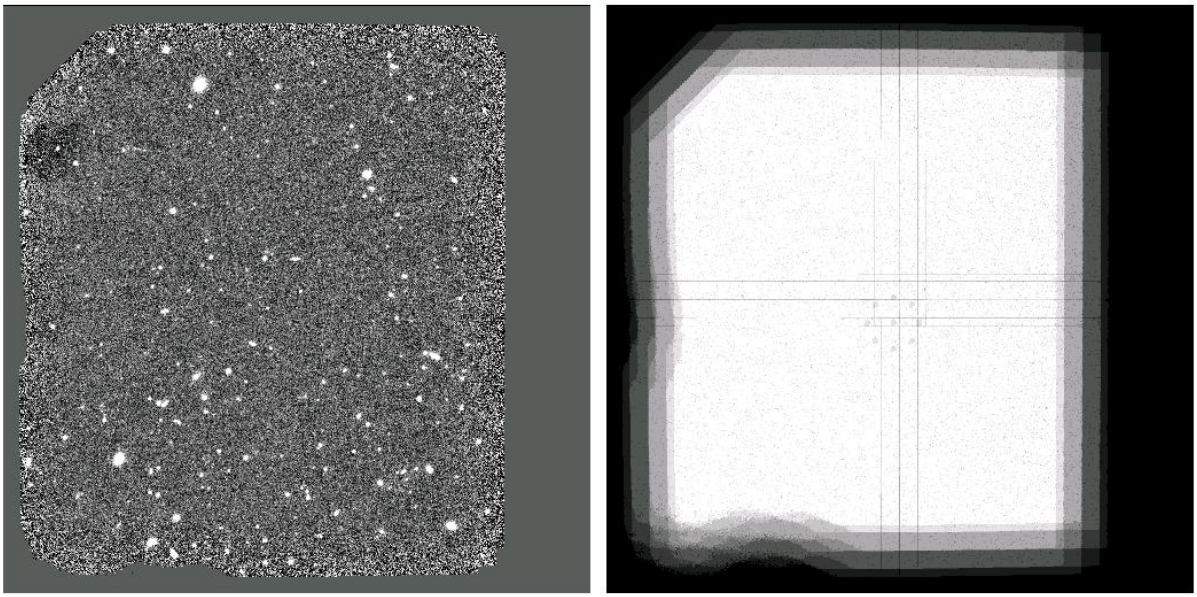


Figure 7: The left hand panel shows the final combined image processed with `mcsall`, and the right hand panel its exposure-time map.

After checking all the parameters, you should be ready to run the pipeline. Here is how to run the pipeline for the data taken under the seeing conditions of 4.5 – 6.0 pixels:

```
mc> mcsall hdfn1_1.lst hdfn1_no1_ch1.fits dosf+ sflat=Selfflat_1_ch1.fits
minfw=4 maxfw=15 thres=4 conn=10 config=dir_mcsred$DATABASE/ana_dec05.cfg
```

As shown above, you do not need to give parameter values in places where you want to keep the default values.

The time for the pipeline to complete all the steps depends mostly on the performance of the computer that you use. For instance, to reduce a data set consisting of 16 frames, it takes approximately 25 minutes for a Linux machine having a CPU with an Intel Xeon 3.4-GHz processor and 4 GB of memory.

Once `mcsall` is successfuly completed, a combined image, as shown in Figure 7, will be created together with logging files produced at each step and an exposure map (see the right panel of Figure 7). The detail of exposure maps will be described in §3.3.3. As for the logging files, see Appendix B for details.

At this stage, we finished reducing the channel-1 data, i.e., those listed in `hdfn1_1.lst`. Subsequently, we suggest reducing another channel data in the same fashion as done for channel 1.

3.3 Mosaicking Images

In order to mosaic the fully reduced images from channels-1 and channel-2, MCSRED offers the task `dmosimg` which combines the two fields observed with the dual-channels into a single image. Although this task is handy for the users who want to mosaic the dual-channel images,

keep in mind that the basic concept of `mcsall` is to reduce each channel's data at first, then mosaic them. Recall that these input files must be distortion-corrected images (see page 10).

The standard procedure using task `dmosimg` is:

1. Supply a list of distortion-corrected images (i.e., up to the Step 6 described in §3.2) as an input of `dmosimg`);
2. Detect objects for image registration using the task `gsextcat` (corresponding to Step 7 in §3.2);
3. Shift-and-add the mosaicked images using the task `gmkgtrimages` (corresponding to Step 8 in §3.2).

As you see above, this task adds a step by `dmosimg` between Steps 6 and 7 of `mcsall`.

To mosaic two images, one must know the "zero-point" of each detector. This yields an intensity scaling factor between the two images for adjusting their flux scales. In `dmosimg`, such a scaling factor can be given by a parameter of `sc`, which corrects the difference of zero points between the detectors:

(sensitivity of channel-1 data) = `sc` × (sensitivity of channel-2 data).

As is clear from the definition, `dmosimg` multiplies `sc` by the channel-2 data. Make sure that you give appropriate correction factors.

Ideally, `sc` should be derived from standard stars data (see §3.5 for how to reduce the standard stars data). As the first order of approximation, `sc` can be estimated from the ratio of the sky-background levels of the dual channels, although the accuracy may be limited by the amount of persistence. An estimate of `sc` is given by $\mathrm{sc} = 10^{[\mathrm{ZP(ch1)}-\mathrm{ZP(ch2)}]/2.5}$ where ZP(ch1) and ZP(ch2) are zero points for the channel 1 and channel 2, respectively.

If you observed with a rather large dithering width, your combined image may have an overlapping region, which may have been observed by the two detectors. Such a common region may be from the right hand edge of the channel-1 field and the left hand one of channel-2. If you are lucky to have a star(s) that falls in the overlapping region, you can use it (them) to estimate the sensitivity ratio.

Here are some examples of the `sc` values for narrow-band filter observations. Notice that these values are estimated from the data reduced with sky-flat and dome-flat frames that were produced by the tasks in MCSRED.

```
# 20060601 – Ks+CSL:   sc=0.813 (under the use of the mcsred flatfield data)
# 20061210 – J+CSL:    sc=0.8913 (under the use of the mcsred flatfield data)
# 20061210 – H+CSL:    sc=0.9120 (under the use of the mcsred flatfield data)
```

3.3.1 Mosaicking task — dmosimg

Task dmosimg combines two fully reduced images taken with the dual detector arrays to a single image. To use this task, users have to give a list of input files, as shown below. The input file should contain a list of images created by mcsall, which are named gcSBsbflMCSA000?????.fits. See Appendix B for the details about files created by mcsall.

```
mc> files gc*fits > gcall.lst
mc> type gcall.lst
gcSBsbflMCSA00014017.fits
gcSBsbflMCSA00014018.fits
gcSBsbflMCSA00014019.fits
gcSBsbflMCSA00014020.fits
....
```

Here, dmosimg assumes that the user always gives data from both channel-1 and channel-2 taken at the same time, and that the contents of the input list file must start with an odd-number file (i.e., channel-1 data) and end with an even-number file (i.e., channel-2 data).

These are the parameters to be set:

```
mc> lpar dmosimg
    inlist = ""              input distortion-corrected file list
     froot =                 the root name for mosaicked individual images
       (sc = 0.8)            Sensitivity scale factor of chip 2.
   (config = "dir_mcsred$DATABASE/ana_feb07.cfg") the name of mcsred config
     (sky = yes)             subtract sky?
     (corx = 0.)             additional dx for ch2 to default mosaic rule
     (cory = 0.)             additional dy for ch2 to default mosaic rule
     (disp = yes)            Display the result?
     (mode = "q")
```

inlist — Give the input file name here, described above.

froot — This parameter specifies "root" name i.e., the common portion of the file names between mosaicked images created by this task. You can give an arbitrary name. For instance, if you specify froot=tmos, the output files will be as follows:

tmos_000?.fits (?=1,2,3,...)
mos_gcall.lst ... output file list (mos_input list name)

sc — This is the parameter to specify a ratio of the two zero points between the two images, as described on page 22.

config — This parameter is to specify an appropriate "configuration" file, as the same as for mcsall (see page 18).

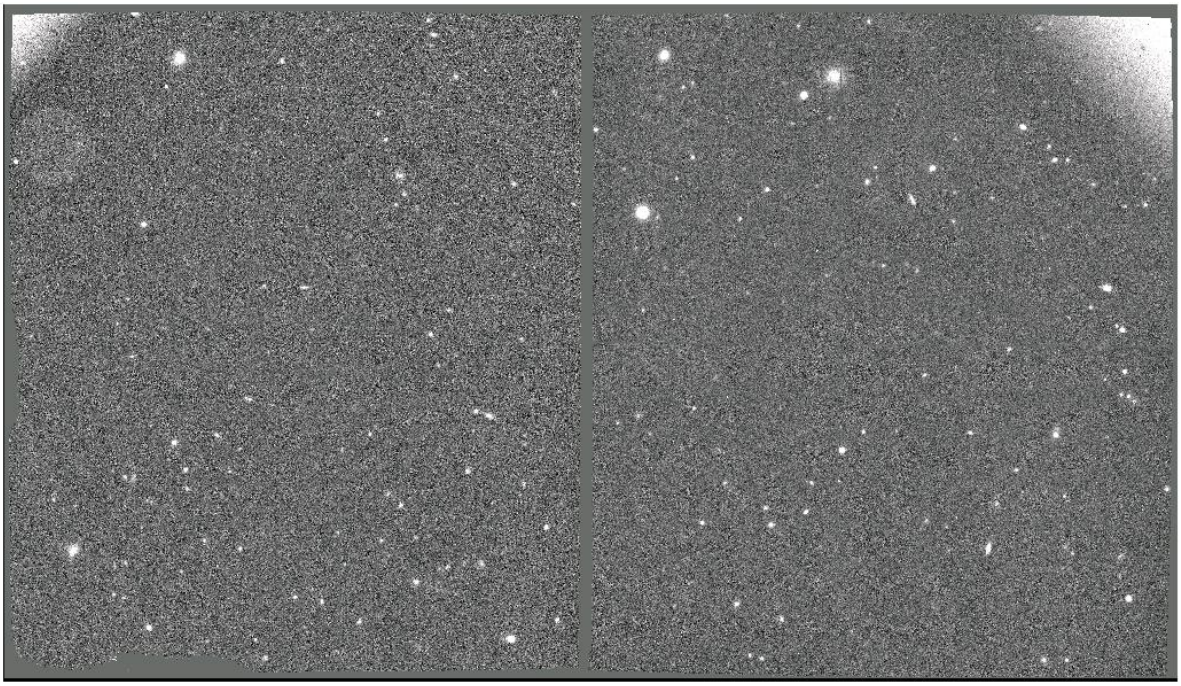


Figure 8: A mosaicked image using `dmosimg` pipeline

`sky` — This is to tell the pipeline whether you want to perform the constant sky-background subtraction. If `sky = yes`, the pipeline calculates the median of the counts over the central region of [700:1300,700:1300], and subtract it from the image plane before mosaicking images.

`corx, cory` — If the images were taken at a low elevation angle, it sometimes happens that mosaic parameters may have shifted from the default values. If you know these offset values, supply them as `corx` and `cory` parameters. If you have star(s) observed in both the channels, you can probably make a good estimate of the offset values.

If you are happy with the input parameters, run the pipeline as shown below.

```
mc> dmosimg gcall.lst tmos sc=0.813 config=dir_mcsred$DATABASE/ana_dec05.cfg
```

If the tasks finish successfully, you will obtain a set of mosaicked images as shown in Figure 8. Here, the default image has a dimension of 3569 $\times$ 2048 pixels with a pixel size scale of 0.117 arcsec pixel^{-1}. Make sure that you gave an appropriate correction factor, `sc`, as `dmosimg` multiplies `sc` to the channel-2 data.

Keep in mind that it sometimes happens that the resulting position accuracy of the mosaicked image produced by task `dmosimg` may degrade if you have tried to combine images taken at elevation angle $\lesssim 45\,\mathrm{deg}$.

If you are not satisfied with the attained accuracy, even if you gave a proper set of `corx` and `cory` parameters, you have to considering doing "accurate astrometry" on these images by giving absolute coordinates as in World Coordinate System (WCS) to each image.

Once you have obtained a mosaicked image, you should next perform the "shift-and-add" process on the images. For this purpose, you have to detect objects in each frame, and shift these images so that each will be overlaid at the same position. In the `mcsall` pipeline, `gsextcat` detects objects in each frame (§3.3.2), while `gmkgtrimages` does the shift-and-add portion (§3.3.3).

3.3.2 Detecting Objects — `gsextcat`

The MCSRED task `gsextcat` detects objects in given frame(s); the parameters you should give are as follows:

```
mc> lpar gsextcat
       infile = "@mos_gcall.lst" Input image name (allow @list forma
       detmin = 12               -DETECT_MINAREA parameter in sextractor (int).
       thresh = 5.               -DETECT_THRESH parameter in sextractor (real).
        minfw = 5.               Minimum size of objects for output catalog.
        maxfw = 15.               Maximum size of objects for output catalog.
        (disp = yes)             Display and check result by display?
       (satur = 15000.)          -SATUR_LEVEL parameter in sextractor.
      (recent = no)              Execute re-centering for SExtractor output?
      (nlimit = yes)             Limit the number of SExtractor catalog objects
        (auto = yes)             Automatically define the detection region
         (xfr = 0.145)           detection avoiding region factor in x
         (yfr = 0.05)            detection avoiding region factor in y
         (cx0 = 296)             Beginning x coordinages for cut-out region
         (cy0 = 102)             Beginning y coordinages for cut-out region
         (cwx = 1456)            x size of the cut-out region
         (cwy = 1844)            y size of the cut-out region
        (mode = "q")
```

`inlist` — Gives a list file of the input images. As shown above, if you want to give a list file, add the @ mark to the head of the file name.

`detmin, thresh` — Gives the DETECT_MINAREA and DETECT_THRESH for SExtractor; these parameters are the same as for ttconn and `thres` described in §3.2.5.

`minfw, maxfw` — These parameters define object size(s) in FWHM to be used in the shift-and-add process. All the objects whose sizes did not fall in the specified size range will not be used, see also `minfw` and `maxfw` in §3.2.5.

`satur` — This is the parameter for SATUR_LEVEL in SExtractor, as described in §3.2.5. Most likely, the default value should be OK.

`disp` — If you want to verify the results using `ds9`, set this parameter to "yes" (the default is "yes").

`recent` — If you want to estimate the central coordinate(s) of the detected object(s), set this parameter to "yes". The default of "no" should suffice for many cases.

`nlimit` — If you set this option "yes", a catalog of the brightest 69 detected objects will be created.

`auto, xfr, yfr` — The first parameter is whether or not to calculate possible coordinate ranges of the objects that will be used for image registration. If you type "yes", the program detects any objects existing in a region defined by (size of images along x-axis) $\times$ `xfr`, and (the same but for y-axis)$\times$ `yfr` centered on the image(s).

`cx0, cy0, cwx, cwy` — If you set "no" for the `auto` parameter above, the pipeline tries to detect objects over a region defined by these four parameters, i.e., [cx0:cx0+cwx-1, cy0:cy0+cwy-1]. Here, `cx0` and `cy0` are, respectively, the x- and y-coordinates of the bottom-left corner of the region, and the parameters of cwx and cwy are the sizes of the region along the x- and y-axes. For instance, if you want to shift-and-add the mosaicked images, you may set `cx0`=100, `cy0`=100, `cwx`=3369, and `cxy`=1848. Notice that all the four parameters are independent of each other, and most likely you can leave them to at default values, namely keeping `auto = yes`, unless you have special reasons.

mc> `gsextcat @mos_gcall.lst 12 5 4 10 auto=yes xfr=0.1 yfr=0.05`

We suggest keeping the total numbers of stars that will be used for shift-and-add to less than 100. This can be controlled by setting appropriate values for the `detmin` and `thresh` parameters, which define the detection threshold, and `minfw` and `maxfw` which specifies the area size to be used. In practice, if you set `nlimit` option to "yes", the numbers of stars to be cataloged in brightness order will not exceed 69. Notice that, if you selected too many (or too few) stars, you may fail in the subsequent shifting procedure.

3.3.3 Shift-and-Add Images — `gmkgtrimages`

The task `gmkgtrimages` will measure (relative spatial) offsets between frames, and combine these images.

```
mc> lpar gmkgtrimages
        inlst = "mos_gcall.lst" The list of input images with associated catalo
       output = "hdfn1mos_no1.fits" Name of result image
      (chkbox = no)             Automatically define the final image size?
        (xbox = 300)            Dither box size in x in pixels
        (ybox = 300)            Dither box size in y in pixels
         (xc1 = 150)            Relative x coord of 1st image in dither box
         (yc1 = 150)            Relative y coord of 1st image in dither box
     (allcomb = yes)            Execute the final combine process?
        (rail = no)             Do the rail remain on chip 1?
     (combine = "average")      Type of the final combine operation
      (reject = "avsig")        Type of rejection
        (zero = "none")         Image sky zero level offset
```

```
    (weight = "sigma")        Weight: sigma or exptime?
    (fitgeo = "shift")        geomap type: SHOULD BE rotate/shift
    (lsigma = 3.)             Lower sigma clipping factor
    (hsigma = 3.)             Upper sigma clipping factor
      (nmat = 40.)            xyxymatch: nmatch parameter
       (tol = 10)             xyxymatch: lolerance parameter
      (nrej = 5)              xyxymatch: nreject parameter
      (sepa = 10)             xyxymatch: separation parameter
       (rat = 6)              xyxymatch: ratio parameter
     (fstop = no)             process will stop after xyxymatch for check
  (fsresume = no)             Resume process using the result by fstop=yes?
      (skip = no)             Skip all processes except the final combine?
      (disp = yes)            display the final result?
    (gtrlst = "")             List of images for final combine
      (mode = "q")
```

`inlist` — Name of the input files list which should be identical to that used for `gsextcat` in §3.3.2 without @ mark.

`output` — Name of the resulting image.

`chkbox` — This parameter tells the program whether or not the image dimension of the combined image should be optimized by the program.

`xbox, ybox` — The numbers of the pixels will be added to the original image sizes in the x-, and y axes. For instance, if you set them equal to the dithering width, you will obtain an image whose dimension should suffice to accommodate all the dithered frames.

`xc1, yc1` — Specify position offsets in pixel units between those for the first image in the input list and the output image. This is because the combined image will have a coordinate system where a position of (X_PIXEL, Y_PIXEL) in the first image corresponds to (X_PIXEL+`xc1`, Y_PIXEL+`yc1`) in the combined image.

`allcomb` — Tells the program if you want to add images; the default is "yes". If you set "no", `gmkgtrimages` will stop just after measuring the offsets between the given frames.

`combine, reject, zero, weight, lsigma, hsigma` — These are parameters for `imcombine`. Select a method of combining from either of `average, median,` or `sum` for the `combine` parameter. The `reject` parameter specifies an algorithm to clip bad data points similar to the so-called "sigma clipping" method. If you have selected this, specify a sigma clipping factor by supplying `lsigma` and `hsigma` parameters. Then, all the data points larger than the `hsigma` value and lower than the `lsigma` will be clipped. If you want to add (or subtract) a constant over the entire image to adjust the sky background level (i.e., median value for many cases), give the desired value in ADU with `zero` parameter. You can combine an image's `weight` by specifying a desired weighting function; `exptime` gives the weighting function on the basis of the integration time of the frames, while `sigma` uses image noise levels.

`fitgeo` — This parameter controls how images will be transformed using the implemented IRAF task `geomap`. Select either "shift" or "rotate"; the former simply shifts images in either the x or y-directions, while the latter includes image rotation. If you are working on data taken under low elevation angles, you can try the `general` option to see if you can attain better image-matching accuracy because such data may require not only shifting but also rotation to obtain a better result.

`nmat,tol,nrej,sepa,rat` — These are parameters for the catalog-matching task `xyxymatch` in IRAF. The defaults will usually suffice. In case the task fails, we suggest changing `rat` to between 6 and 9, or/and changing the `tol` and `sepa` parameters. See the help page for the IRAF task for the details.

`fstop, sfresume` — If you want to stop the pipeline after making a catalog-matching database for position corrections, set `fstop = yes`. As mentioned above, if you fail in catalog-matching and you need to search for appropriate parameters, try with `fstop = yes` to obtain the best parameters. Once you succeed in finding such parameters, execute the pipeline with `fstop = no` and `sfresume = yes` to let the pipeline finish combining position-corrected images.

`skip` — If you select `skip = yes`, `gmkgtrimages` skips the shift-and-add process and tries to combine (final) image. This function will be handy if you have already run `gmkgtrimages` once before, and want to change or test an algorithm of combining and/or clipping bad pixels. If this is the case, keep in mind that you must supply the list of images whose positions have been shifted appropriately. In addition, if there are images that you don't want to include when combining final image, don't forget to edit the input file list given in `gtrlst`, and set `skip = yes`. For instance, if you wish to compare results from the "image-weighting" algorithm described above with that from the so-called "median algorithm", you may try to execute the command described below. Notice that before issuing the command below, you must have executed the step `gmkgtrimages` without any problem.

```
mc> gmkgtrimages mos_gcall.lst MOSRESULT2.fits comb=median reject=minmax
gtrlst=GTRmos_gcall.lst skip+
```

`gtrlst` — Gives a list file of the position-shifted images that are ready to combine. For many cases, the user should specify the file of "GTR" which must should have been created in the prior successful execution. Note that the "GTR" file is the one you specified with the `inlst` parameter.

Note that intensity scales of all the images (i.e., both the final combined image and position-shifted images before combining) produced by `gmkgtrimages` are normalized to counts per 1-second integration (See §A.7 for details). Figure 9 shows an example of the final combined image. In addition to the above, the pipeline will also produce position-shifted images before combining, and bad pixel masks for each frame, as well as the following log files:

- `gmp`input_list.`log` – a logging file, in the case of the above, `gmpmos_gcall.lst`

Figure 9: The final combined image mosaicked by task `gmkgtrimages`.

- `exp`_outputimagename.`pl` — an image of the exposure map whose intensity in ADU units corresponds to 1 second integration, e.g., `exp_MOSRESULT2.fits.pl`
- `sgm`_outputimagename.`fits` — the so-called "sigma image" which represents standard deviations at each pixel position over all the images before the combining, e.g., `sgm_MOSRESULT2.fits`.
- outputimage.`pl` — A data file which stores bad-region-mask data. The name is `MOSRESULT2.pl` in the above case. If the next integration time at the masked areas become less than half of the non-masked regions, the data from these areas will be masked.

3.4 Limitations of `mcsall`

We stress that users should bear in mind that `mcsall` does not take into consideration the quality of each of the frames (except for their noise levels) when combining the final image. If you are not happy with your final result, we strongly suggest verifying such variables as seeing sizes and variation of atmospheric transmissions, which may alter not only stellar sizes but also total counts. If you identify such frames, try to correct them (if possible), or it would be prudent of you to not use such frames.

It would be useful to inspect the total counts and seeing sizes of several stars between frames before combining them into the final image. If you find obvious discrepancies in some frames, you may have to multiply them by scale factors and/or correct their PSFs. MCSRED does not offer such data inspection and correction functions owing to the practical reason that it is difficult to expect high-enough numbers of bright stars that would allow such corrections AT ANY TIME. Therefore, we suggest always taking a conservative approach if you want to attain high accuracy in photometry. In this context, we would say that MCSRED can give

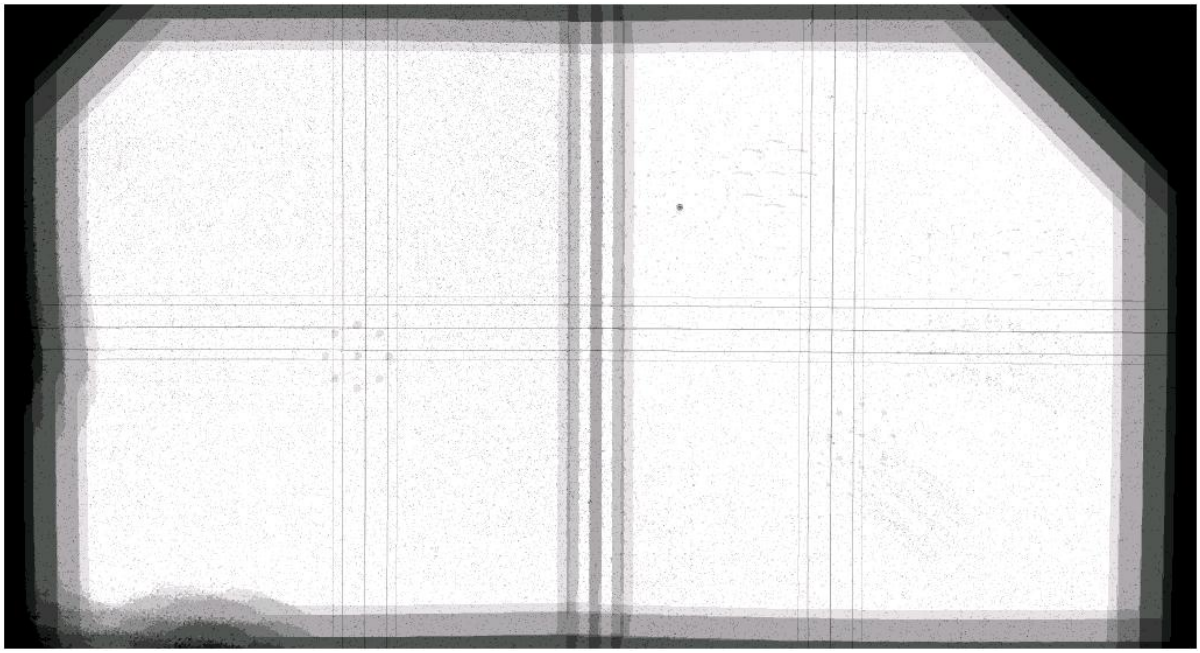


Figure 10: An example of exposure-time map for a mosaicked image.

only an averaged image as its final combined image.

For instance, let's consider the accuracy of the flux density scale of a final image. Many observers have observed standard star(s) before or after observing target field(s). If sky conditions had been stable through a set of observations, the observer would obtain reasonable results by applying a flux scale conversion factor estimated from standard stars data. However, if a portion of the data was taken while atmospheric opacity had varied significantly, you cannot use a zero-point value obtained from the standard star data that were taken at a particular time of the observation. In other words, you must take into consideration the time variation of the zero point throughout the observing time.

In conclusion, we strongly recommend that as you use MCSRED and `mcsall`, you consider all the possible limitations attributed to the instrument, the package itself, observing conditions, and that you keep scientific goals in mind.

3.5 Reducing Standard Stars Data

In order to convert the intensity scale of images from count in ADU sec^{-1} to magnitude, we assume that you have observed standard stars whose magnitudes are well known. Once you measure numbers of the incident photons per second towards a standard star, you readily have a conversion factor between magnitude and count per second (ADU sec^{-1}) which is referred to as photometric zero point. Unless you want to attain a very high accuracy in photometry, it will suffice to observe a standard star once immediately after or before observing your target at an airmass as close to the target as possible.

Since the majority of standard stars are very bright, they are usually observed over a short integration time so as not to saturate the detectors, for example, defocusing the stellar image or using a partial readout. If you observed one or more standard stars with defocusing, you can use `mcsall`. On the other hand, if you observed the standard star(s) with partial readout, you cannot use `mcsall`, although each task implemented in MCSRED helps you. Therefore, we describe a method reducing partially read-out standard stars data in §3.5.1. Finally, as of 2009, the partial readout towards standard stars is avoided except in a few cases.

3.5.1 Partial Readout

Since MOIRCS does not have a shutter, the minimum exposure time is determined with the readout time of the detectors. The minimum time is currently limited to 21 seconds under the standard "whole readout with dummy". However, it is possible to complete your exposure in less than 21 seconds by reading out data partially, i.e., partial-readout. Namely, you can finish your integration within a shorter time than that required to read all the pixels; partial-readout usually reads the pixels in arrays of 1024×1024 or 512×512 centered on the field-of-view. To check if your data were obtained through partial-readout, see the keywords of `PRD-MIN[12]` and `PRD-RNG[12]` in the FITS header. In practice, we found problems in the partial-readout process in 2008 November. We have been trying not to use it since then.

3.5.2 Reducing Standard Stars Data

All the standard stars data described here were obtained by means of whole-readout. One can reduce them using `mcsall` (as for partial-readout, see §3.5.1). The most important precaution to remember when you are reducing standard stars data is that **you cannot use any defocused images to combine an image.** Keep in mind that stellar images in the defocused frames do not show identical morphology. If you combine such images, you will lose or add counts to the star(s). With this reasoning, we strongly suggest performing photometry of defocused images **one by one**, and adopting their mean to have a decent (and perhaps better) estimate.

Given the above, you must not process further steps beyond sky-subtraction (or distortion correction) when you reduce standard stars data using `mcsall`. Specify `bye = 6` to stop the pipeline before detecting objects (see §3.2.7). Another caution is that **you MUST always use exactly the same flat frame that was used for scientific frames.** If you mix flat frames, it will degrade accuracy in flat-fielding, especially for the cases of short-exposure

time observations. In this subsection, we describe how to reduce standard stars data taken on December 9, 2005 as an example. We start by reading the header list created in §3.2.4.

```
mc> less header.txt

MCSA00014102    HDFN1   KS      100.000
MCSA00014103    HDFN1   KS      100.000
MCSA00014104    HDFN1   KS      100.000
MCSA00014105    HDFN1   KS      100.000
MCSA00014106    HDFN1   KS      100.000
MCSA00014107    FS33    KS      30.000
MCSA00014108    FS33    KS      30.000
MCSA00014109    FS33    KS      30.000
MCSA00014110    FS33    KS      30.000
MCSA00014111    FS33    KS      30.000
MCSA00014112    FS33    KS      30.000
MCSA00014113    FS33    KS      30.000
MCSA00014114    FS33    KS      30.000
MCSA00014115    TWILIGHT        KS      13.000
MCSA00014116    TWILIGHT        KS      13.000
```

The above tells that a standard star of FS33 was observed immediately after observing HDFN. FS33 is one of the well-known standard stars listed in "UKIRT Faint Standards" (Leggett et al. 2006, *MNRAS*, **373**, 781); the star is often observed as a standard star in NIR imaging observations to estimate before photometric zero point of intensity scales.

Since MOIRCS has two detector arrays, we strongly recommend observing a standard star with the two array channels using the following procedures: (1) observing a selected star with a detector channel, (2) observing the star at different sky positions (i.e., pointing positions of the telescope) so that the star falls at a different pixel position on the array, and (3) repeating these two procedures with another channel. The header information tells that the standard star, FS33, was observed in the four frames between MCSA00014107 and MCSA00014114. Check these images to see that the star is seen in channel-2 in the first two frames, and in channel-1 in the last two. Make a list of the frame-IDs to be used in `mcsall`.

```
mc> listprep /home/analysis/FITS/FS33 14107 14114
```

We also suggest reducing another detector channel where the standard star was not observed (a blank sky only for many cases) in the same manner because this is useful for sky subtraction. Just in case, make sure that the standard star's data were obtained by means of whole-readout because `mcsall` cannot be used for partial-readout data.

```
mc> hselect @FS33_1.lst FRAMEID,PRD-RNG1,PRD-RNG2 yes

MCSA00014107    2048    2048
MCSA00014109    2048    2048
MCSA00014111    2048    2048
MCSA00014113    2048    2048
```

Here, the outputs for the parameters of PRD-RNG1 and PRD-RNG2, respectively, indicate ranges of the pixel coordinates along the x- and y-axes used for partial readout. The two values of 2048 mean that the data were read out over the whole pixel area.

Once you checked that the data were not produced by partial readout, execute `mcsall`, as follows:

```
mc> mcsall FS33_1.lst FS33_ch1.fits dosf- extflat=Selfflat_1_ch1.fits bye=6
```

Use the flat flame(s) that were used for scientific flames by setting `dosf = no` and giving the name of the flat frame(s) in the `extflat` parameter. In the above example, we force the pipeline to stop at Step #6, i.e., after distortion correction, by specifying `bye = 6` before performing image combining. Unless your standard star(s) have been observed at the gap(s) between the arrays, you can stop at the Step #5, though the distortion correction is not crucial. If this is the case, the Step #6 may help you to deal with such data accordingly.

If you are happy with the results, execute `mcsall` on the other channel data to complete your data reduction, as follows:

```
mc> mcsall FS33_2.lst FS33_ch2.fits dosf- flat=Selfflat_1_ch2.fits bye=6
```

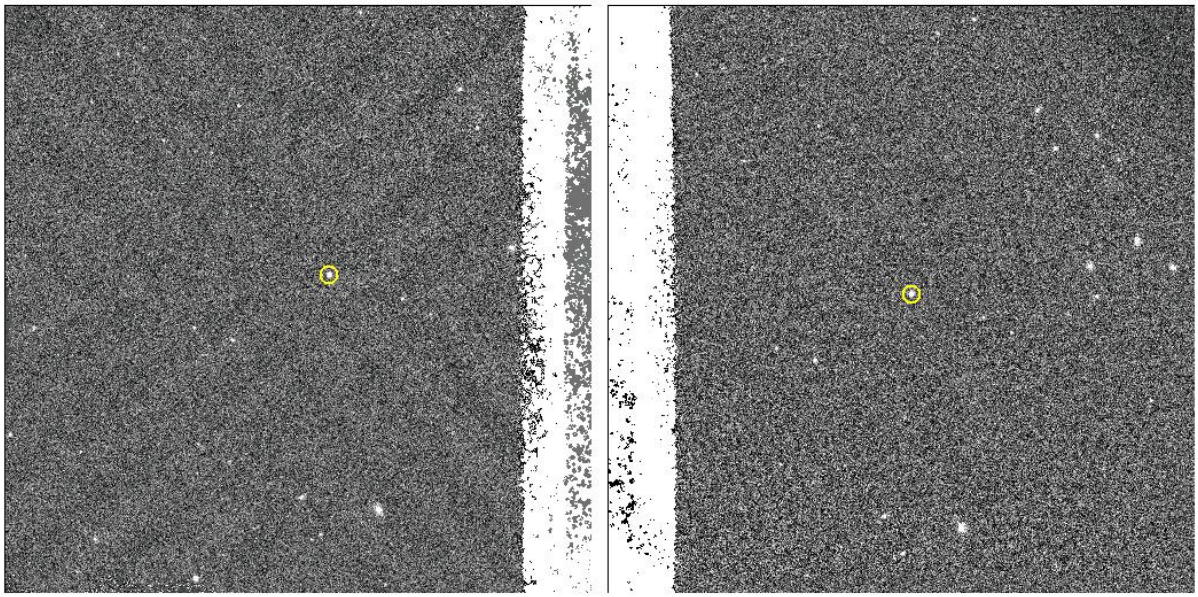


Figure 11: An example of sky-subtracted data including a standard star (left: channel-1 data `gcSBsbflMCSA00014111.fits`; right: channel-2 data `gcSBsbflMCSA00014108.fits`). The object enclosed by the yellow circle is the standard star, FS33.

Did you obtain images shown in Figure 11? If so, you have successfully reduced your standard stars data up to the sky subtraction process.

Next, we describe a "quick" method to estimate the zero point. According to the "UKIRT Faint Standards", the standard star FS33 has a K-band magnitude of 14.296 ± 0.011. We

therefore need to know how much count was measured per second by MOIRCS towards the star to find the desired zero point.

Display the channel-1 images as follows:

```
mc> display gcSBsbflMCSA00014111.fits 1 zr-- zs-- z1=--100 z2=150
```

Use the IRAF task `imexamine` to measure the standard count. If you are using IRAF ver. 2.12. or later, the task automatically finds the most appropriate aperture size. Execute the task by entering the command,

```
mc> imexamine
```

You will see a blinking circular cursor on your image viewer, e.g., `ds9` (see Figure 11). Type 'a' on the circle and the following will appear on your terminal:

```
#   COL    LINE     COORDINATES
#     R     MAG     FLUX      SKY     PEAK     E    PA BETA ENCLOSED   MOFFAT DIRECT
1132.68 1109.94 1132.68 1109.94
  15.71  19.01 335871.    1.91   8265. 0.02  -50 6.88     5.26     5.36   5.24
```

Here, "FLUX" is the measured count towards the star, and "R" the aperture radius used. Recall that the data were taken with a 30-second exposure (see `header.txt`). The desired count per second is obtained from dividing the "FLUX" by the exposure time, i.e., 335871/30 = 11195.7 ADU sec^{-1}. Since this should correspond to the brightness of $K = 14.296$, the zero point is estimated by

$$\mathrm{ZP(ch1)} = 14.296 + 2.5 \times \log_{10}(11195.7) = 24.42$$

Similarly, one can estimate a zero point value for the star measured at channel-2. In the case of the sample data, the channel-2 image is "gcSBsbflMCSA00014108.fits". We should obtain

```
#   COL    LINE     COORDINATES
#     R     MAG     FLUX      SKY     PEAK     E    PA BETA ENCLOSED   MOFFAT DIRECT
1058.47 1042.94 1058.47 1042.94
  16.07  18.79 410440.   2.842   9299. 0.07   67 3.49     5.38     5.48   5.36
```

One can convert the obtained values into the zero points as

$$\mathrm{ZP(ch2)} = 14.296 + 2.5 \times \log_{10}(410440/30) = 24.64$$

Once you know these zero points, you can convert the total count of the object(s) into a flux density scale. We strongly suggest reading appropriate textbooks to learn methods of photometry more accurately. It is also good idea to check the sensitivity ratio between the

channels as follows

$$\mathrm{sc} = 10^{[24.42-24.64]/2.5} = 0.817$$

which agrees with the value introduced in §3.3.1.

You may want to set an aperture radius by hand (i.e., not using the automatic function). For instance, you may want to measure the intensity of the standard stars within a given small radius for images taken at defocused positions. If this is the case, you have to select a radius so that the mean of the stellar counts becomes close to zero (see Figure 12). After finding such a radius, you have to tell `imexamine` the radius to perform photometry. We describe an example below using the same data.

First, you have to stop the automatic procedure to find the center of emission.

mc> `rimexam.center=no`

Second, evaluate the radial profile of the stellar counts to find a radius where the star cannot be recognized.

mc> `rimexam.rplot=40`

Third, execute `imexam` and type `r` on the star. This gives you a plot showing the radial intensity profile of the target star up to a radius of 40 pixels (see Figure 12).

Check the displayed intensity profile to find a radius and counts where the star and the sky background level cannot be distinguished. Quite the task by `q`, then,

mc> `rimexam.iterations=1` (this option turns off the automatic determination function of aperture radius)
mc> `rimexam.radius=15`
mc> `rimexam.buffer=5`
mc> `rimexam.width=10`

Here, `radius` is a radius to define a circular aperture over which the count of the star is measured. The parameter `buffer` is a separation between the `radius` and an inner radius of an annulus to measure the sky background level. The outer radius of the annulus should be given by `radius`+`buffer`+`width`. In the above example, we will measure the stellar count over a circular aperture whose radius is 15 pixels. The sky background level is estimated over an annulus whose inner and outer radii are, respectively, 20 and 30 pixels, and will be subtracted from the stellar counts. Once you select radius, execute `imexamine` once more, move the cursor onto the star, FS33, and type '`a`' on it to perform photometry of the star.

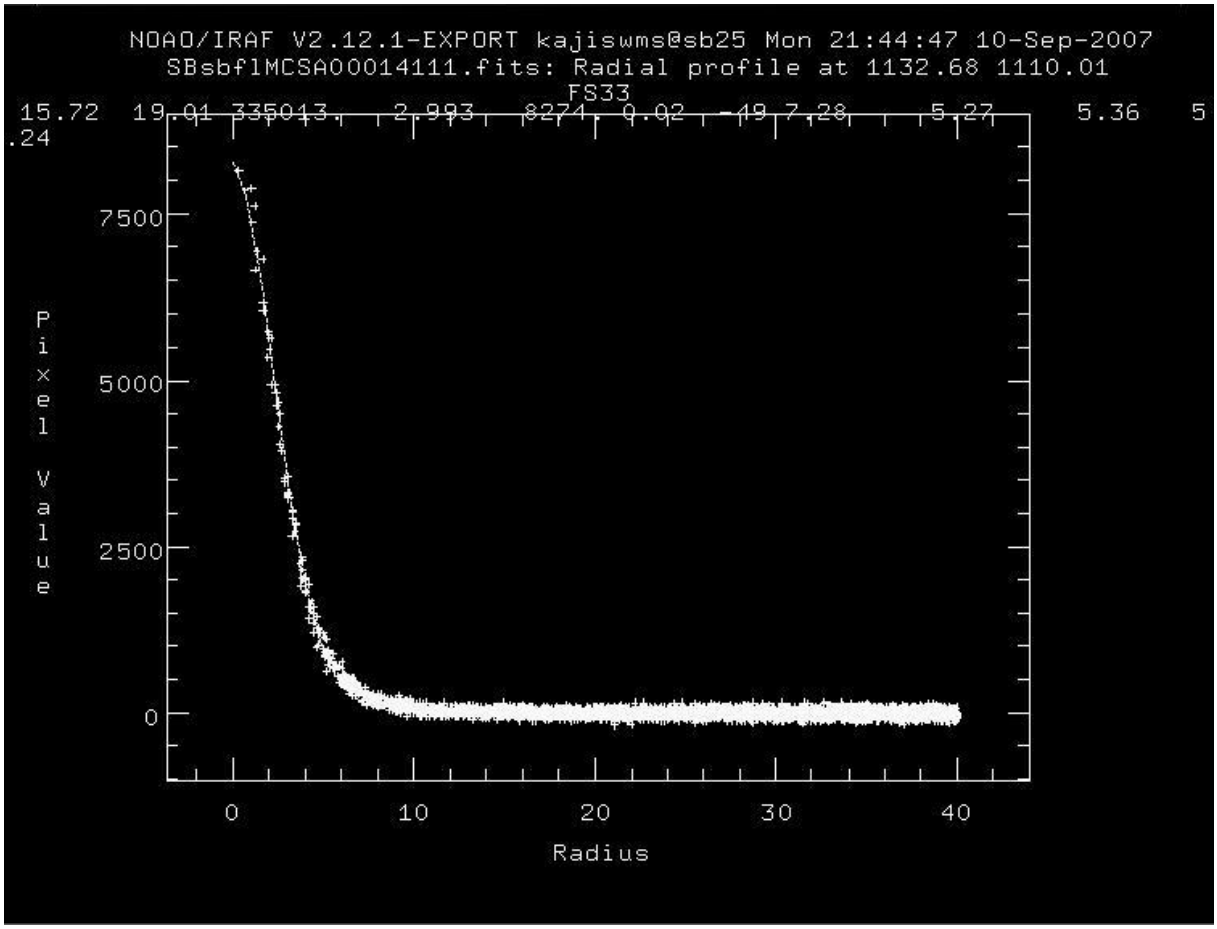


Figure 12: An example of the radial intensity profile of a standard star. In this particular case, we explicitly specified a range of radius to be displayed when we executed the IRAF task `rimexamine`.

After completing photometry, we suggest clearing all the parameters to the default values by typing,

```
mc> unlearn rimexamine
```

We also strongly suggest reading the help page for the task `apphot` and the associated documents. For your convenience, a document written in Japanese by Dr. F. Yoshida can be found at

`http://subarutelescope.org/Observing/DataReduction/mtk/subaru_red/autumn07/spcam_lec.pdf`

3.5.3 Reducing Partial Readout Data

In this subsection, we describe a method for how to reduce data sets taken by partial-readout. As briefly mentioned in §3.5.1, partial readout is a technique to shorten exposure time, and is sometimes used for observing (very bright) standard star(s) (see Figure 13). Recall that `mcsall` cannot deal with partial-readout data. Instead, MCSRED offers tasks called `prmask` and `cutpr` to reduce partial readout data; the former makes a mask frame when detecting

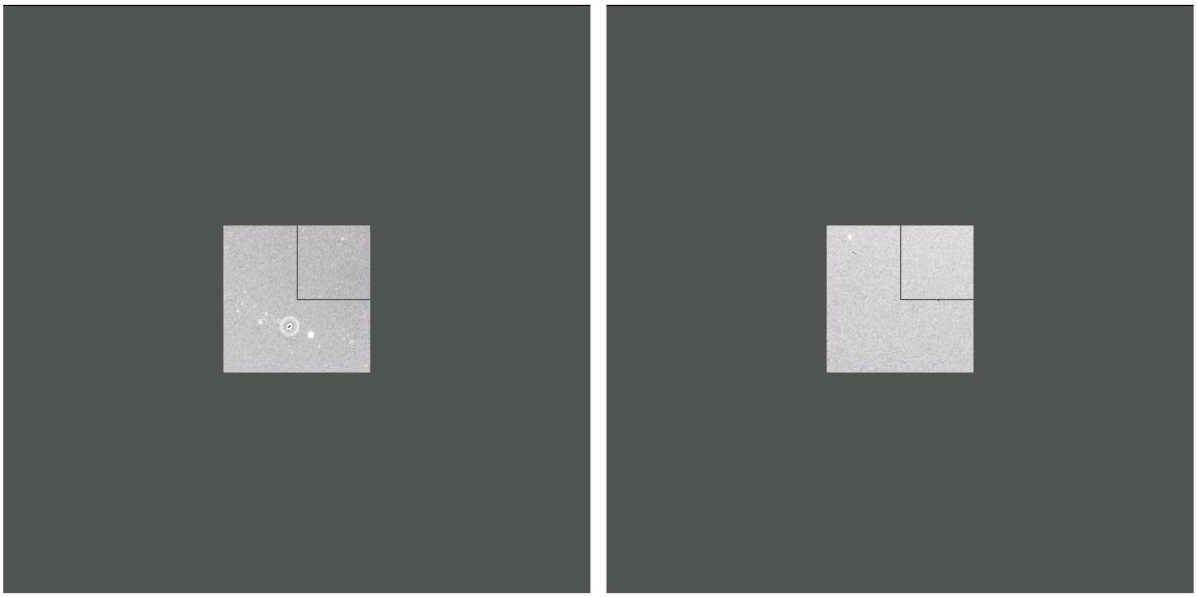


Figure 13: An example of partial-readout images (left: channel-1, right: channel-2) where the central regions of 512 × 512 pixels were being read.

objects, and the latter extracts a sub-image using only the pixels where the counts were read.

Below we summarize an outline of reduction for data that have been partially readout:

1. Make an object mask using the `prmask` task.
2. Flat-fielding; don't forget to use the same flat frame that was used for your scientific object(s).
3. (Median) sky-subtraction using `sbselfsky`.
4. Extract an image only for the pixel regions where counts were read out using the task `cutpr`.
5. If necessary, subtract the sky subtraction residual using the task `tsubanomaly`, instead of using `qmsepskysb`.

Here, `tsubanomaly` is a task – corresponding to `qmsepskysb` — to subtract the sky-background level valid for the images whose dimensions are different.

When you apply a distortion correction, use `mcsgeocorr` before making a sub-image using `cutpr`. In practice, we believe that almost all observers will not apply distortion corrections to partially readout data because partial-readout is used mainly for standard stars.

Below is an example using the images under `/RAW`. Make a list file as

mc> `listprep /RAW fs103 39115 39126`

Give the obtained results to `prmask`; the default parameters should be OK for many cases.

```
mc> prmask fs103k_1.lst|
mc> prmask fs103k_2.lst
```

The next step is flat-fielding using the IRAF task `imarith`. Here, you should check the list files `bisfs103k_?.lst` where ? is either 1 or 2 under the current directory; the list files are created by `prmask`, and store names of the raw data files. Using the lists, let's make a list of images after flat-fielding. Below, we make such a list using `awk` on Linux.

```
mc> !awk 'print "sf"$0' bisfs103k_1.lst > flbsfs103k_1.lst
mc> !awk 'print "sf"$0' bisfs103k_2.lst > flbsfs103k_2.lst
```

Check the contents of the lists which will be used for flat-fielding. Notice that we should divide the standard star's images by the flat frame, i.e., `SFlat_ch?.fits`, that were used for your scientific (target) frames.

```
mc> imarith @bisfs103k_1.lst / SFlat_ch1.fits @flbsfs103k_1.lst
```

This will create output images (flMCSA****.fits) under the directory where you issued the command.

The next step is median-sky-subtraction. Give the name of the ASCII text file which describes names of the images that have been flat-fielded to the task `sbselfsky`.

```
mc> sbselfsky flbsfs103k_1.lst 0
mc> sbselfsky flbsfs103k_2.lst 0
```

Here, we set the `nself` parameter to be 0, which tells the pipeline that you want to perform "median-sky-subtraction" for all the data given in the list, except for the image frame from which the median-sky will be subtracted.

After completing this stage, you are ready to perform photometry of the standard star(s). However, your image may have a pinwheel (windmill)-like pattern in its low-level counts. You probably want to remove such a pattern to improve accuracy in photometry. Below we offer a method to remove such a pattern. Notice that we have to work on all the images where the star is recognized, and apply the procedure below to each image. Repeat these procedures until you have done all the images.

To remove the pinwheel pattern, we cut the partial-readout portion(s) of the images using `cutpr`, as follows:

```
mc> cutpr sbsfMCSA00039121.fits ctsbsfMCSA00039121.fits
```

The resulting image should be given to task `tsubanomaly` as an input; the task should help you to remove the residual pattern.

```
mc> tsubanomaly ctsbsfMCSA00039121.fits order=5
```

We believe that you don't need to use a high order; 3 to 5 should be OK for many cases. Check the output image of `SBctsbsfMCSA00039121.fits` with `ds9` whether it is flatter than before.

3.6 Cleaning Up Your Account — `cleanall`

As you may have realized, `mcsall` creates various files (see Appendix B for details). Some of them are intermediate-products which users do not necessarily need to save. You can delete them with the cleaning-up task, `cleanall`. For instance, give a list file for the files you want to clean

```
mc> cleanall hdfn1_1.lst level=0
```

where the `level` parameter specifies what kind of (temporary) data you want to save (or discard), as shown below:

- level=0: Making a tar archive of all the files. All the images, except for the raw data, will be stored in the tar file.

- level=1: Saving sky-subtracted images, mask frame(s), object catalog(s), and the files used for shift-and-add. If you specify `level=1`, `cleanall` will delete all the files which can be easily reproduced.

- level=2: Keeping the final image and all the list files, and deleting all the others.

4 Procedures Not Implemented in `mcsall`

In this COOKBOOK, we describe a series of "standard" data reduction steps that are implemented in `mcsall`. However, some tasks which are not performed in `mcsall`, but provided in the MCSRED package may help some users whose data are not properly dealt with in such standard procedures. In this section, we give a brief explanation about additional tasks.

4.1 Reducing Data Acquired with NODDING

"Nodding" is a technique to minimize the effect of atmospheric conditions for observing an extended object (emission). If you are not sure whether or not you have used nodding, check if you observed with the `SKYNOD` option for `GETOBJECT` command. Different from dithering observations, your observations were probably done with a sequence of

$$\text{Object} \longrightarrow \text{Sky}\,1 \longrightarrow \text{Object} \longrightarrow \text{Sky} \longrightarrow \text{Object} \longrightarrow \ldots$$

You may have repeated a cycle of observing your target field(s) and getting sky frame(s). To reduce such a data set, MCSRED offers a package of `noddata` which includes the following tasks[§]:

- `subskyimage` — Making an object-mask frame for sky frames, and performing both sky-background subtraction and flat-fielding
- `gcrsbimg` — Correcting distortion for single-channel data
- `tiltskycor` — Subtraction residual of sky-background subtraction, i.e., removing a pinwheel pattern only
- `mosgcsbimg` — Mosaicking distortion-corrected images using `gcrsbimg`
- `mossbimg` — Performing both distortion correction and mosaicking at the same time for the dual-channel data, i.e., this task is identical to `gcrsbimg` plus `mosgcsbimg`.
- `skycorgmk` — Re-combining sky-subtracted images which are produced by `gmkgtrimage`

Below describes an overall set of procedures for data taken with nodding observations.

1. Inspect header parameters such as integration time, using, e.g., `hselect` in IRAF. Notice that `subskyimage` assumes that all the input frames have an identical integration time.
2. Make list files for the dual channels using the `subskyimage` task.
3. Sky-subtraction, flat-fielding, and masking of objects in sky frames using the `subskyimage` task.

[§]The `noddata` package may not work properly for the data taken before 2007, due to improper description of some header keywords.

4. Subtract the residual sky background emission from the above, i.e., removing the pinwheel pattern (see §A.3) for many cases.

5. If sensitivity ratio between the dual channels is:

   (a) known, execute `mossbimg`.

   (b) not known, or you do not need to mosaic the dual channel images, proceed distortion correction with `gcrsbimg`. If you want to mosaic images further, use `mosgcsbimg`.

6. Detect stars with the task `gsextcat` for shift-and-add, as done for dithering observations.

7. Execute shift-and-add with `gmkgtrimages` in the same manner as done for dithering observations.

8. Combining all the frames by adjusting image-background levels with `skycorgmk`.

For your information, MCSRED offers a pipeline task, `nodmosall`, that executes all the tasks above. If you are interested in it, read `MOIRCS_SKYNOD_REDUCE.txt` under the `MCSRED/` directory.

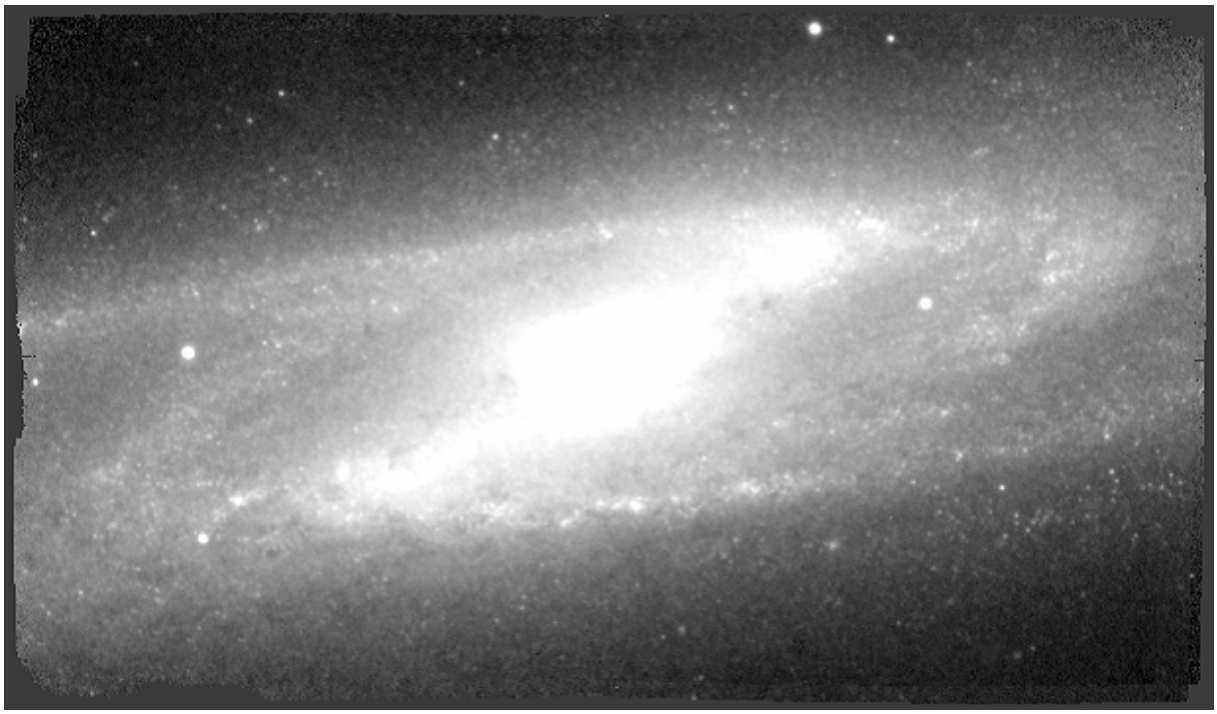


Figure 14: An example final image generated with `noddata` package

4.2 Making an Object Masking Frame from the "Final" Images

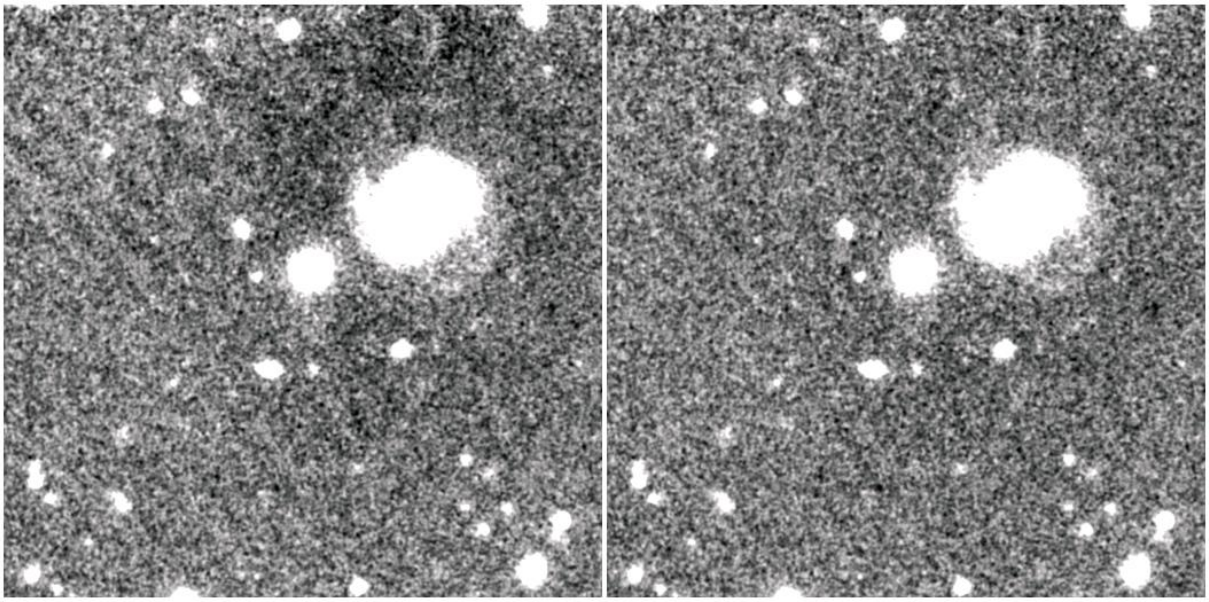


Figure 15: The left hand panel represents a tentative "final" image combined with `mcsall`, while the right hand one shows the final image reproduced using an object mask which is made from the tentative "final" image shown at left.

Using task `mcs_mksflat` in `mcsall` pipeline, we made an object mask by detecting potential objects on each of the raw data frames. However, if we use the final combined image, we can detect much fainter objects than those detected in the individual raw frames. If you want to attain a higher accuracy in sky-background subtraction, we suggest trying to make another object mask frame by masking fainter objects which are not recognized in each frame; this should contribute to flatten the sky-background level of the final image (see Figure 15). If you have very bright object(s) in your FOV and/or if you performed very deep long-integration observations, this is worth trying.

For this purpose, we have to complete making the (tentative) "final" combined image. Using the image, we should make an object frame which should be used for flat-fielding and median-sky-subtraction processes when you repeat the whole data reduction pipeline. As of October 2009, MCSRED offers the `invmask` task, which creates another object mask from such a combined image[¶]

The typical procedure for reducing data using `invmask` is as follows:

1. Run `invmask` under the directory where you have executed `mcsall` once. This is to make new mask-frames (`objmsk000?????.pl` where ????? is frame IDs).
2. Move both raw data (e.g., MCSA000?????.fits) and the newly created object mask (i.e., objmsk000?????.pl) to any other directory (making a new directory would be better). Here, these files can be found under the directory where you executed `mcsall` and `invmask`.

[¶]If you are using IRAF v2.12.1 or earlier, see §3.1.3.

3. Execute `mcsall` under the new directory

This will (hopefully) improve the accuracy of the sky background subtraction of your image(s).

4.3 Removing Fringe Patterns

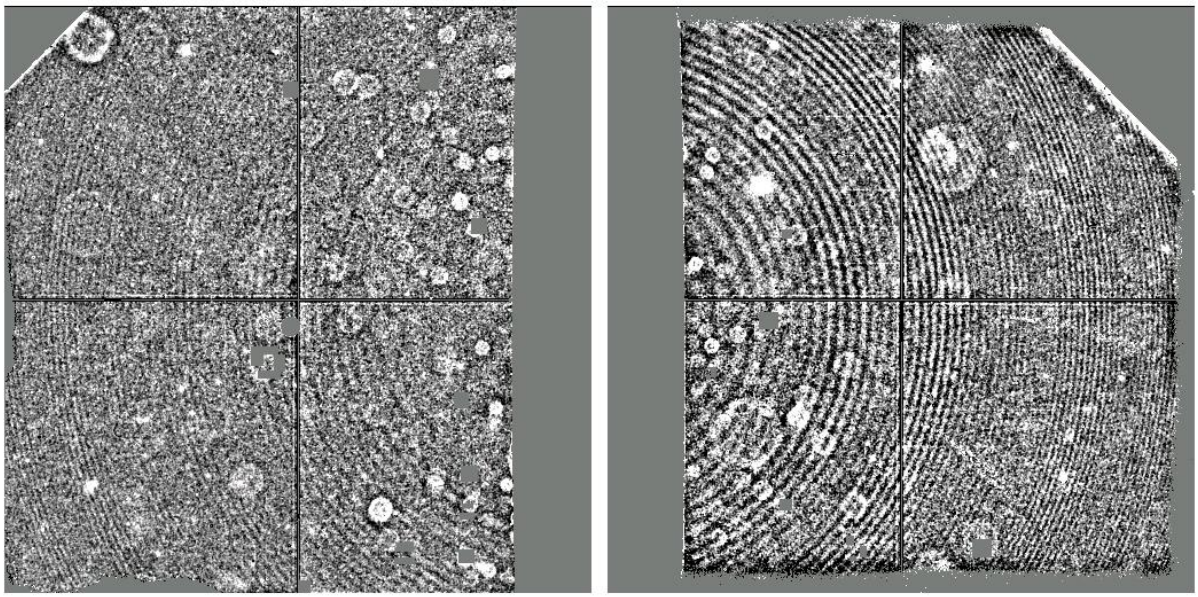


Figure 16: Examples of fringe patterns seen in MOIRCS data. The two images are obtained from dividing raw-data (left: MCSA00014017.fits, and right: MCSA00014018.fits) by dome-flat, and subtracting sky background emission. To enhance the fringe pattern, we further smoothed these images by convolving with a Gaussian image whose HPBW is 2 pixels. The small (mostly white) circles are due to dust or/and bad pixels.

In principal, MOIRCS data always contain a fringe pattern which can be recognized if you divide an object image by dome-flat as shown in Figure 16. This is probably the most recognizable in channel-2 data in the K-band. These fringe pattern may be removed in the median-sky-subtraction process for many observations if sky conditions were stable while the frames for median-sky had been taken. In addition, it should be noted that sky flats may also have similar low-level (1-2%) fringe patterns.

MCSRED does not have a task to remove the fringe pattern. However, an IDL package written by Dr. W. Wang does have a task that removes such patterns. The IDL package is free to obtain from the MOIRCS webpage. Another possibility is to use an IRAF task written by Dr. M. Kajisawa, the author of this COOKBOOK (in Japanese). Since his task is not implemented in MCSRED, those who are interested in it should contact him. Here, we just present an example how such a fringe pattern is being removed (see Figure 17).

As of 2009 July, MOIRCS is equipped with a filter so as not to cause fringe patterns. See the MOIRCS web page for details.

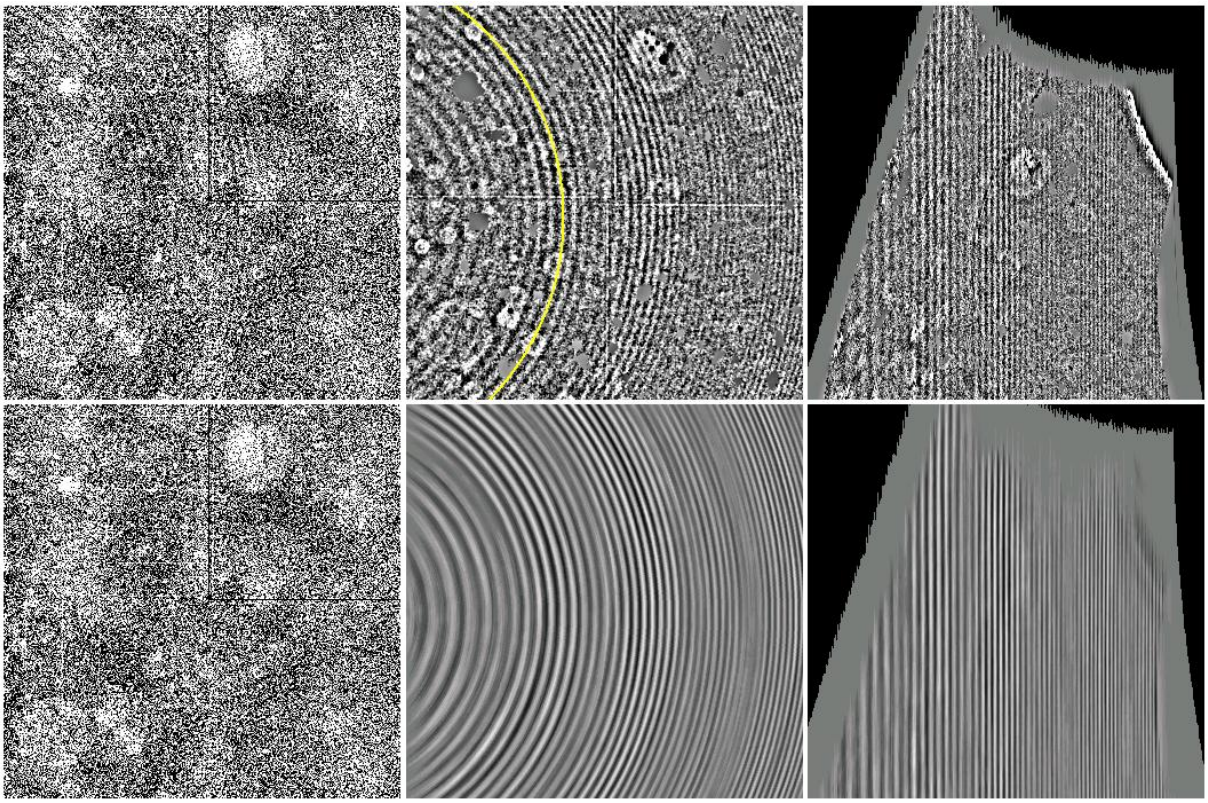


Figure 17: An example of removing a fringe pattern seen in MOIRCS data. The upper left panel represents an example image that is obtained by dividing raw data (upper left) by the dome-flat and the subtracted sky-flat. The upper center panel is obtained by masking objects and smoothing with a Gaussian to enhance the fringe pattern. Using the upper center panel, one has to find the center of the fringe-pattern. The identified pattern is converted into another plane defined by its radius vs. position angle — see upper right panel where the horizontal and vertical-axes are, respectively, the radius and position angle. Subsequently, we apply a median filter to the converted image to obtain another image shown in the lower right panel. If the result is OK, we should transform the image back to the original coordinate system (lower center panel), which should be subtracted from the very original image (upper left). The bottom left panel shows the resulting image, which is free from fringe pattern.

A Tasks Used in `mcsall`

In this section, we describe general notes about tasks implemented in the `mcsall` pipeline. If you are interested in more details, see the README file that can be found in the `MCSRED` directory.

A.1 `mcs_mksflat`

The task `mcs_mksflat` allows you to make a self-sky-flat frame. Even if you specified that you do not want to apply self-sky-flat in subsequent processes (i.e., `dosf = no`), when you execute `mcsall`, `mcs_mksflat` will be executed in the background to make an object mask that will be used in subsequent steps, e.g., median-sky-subtraction. Figure 18 shows an example of a sky flat.

Notice that users do not perform dark subtraction for many cases, as of December 2005, because contribution from the dark current at MOIRCS is negligibly small.

The standard procedure is:

1. (Subtract dark.)
2. Make an object mask:
   (a) Dividing each frame by its subsequent frame
   (b) Subtracting sky-background emission in each quadrant of the detector arrays using a lower-order fitting function.

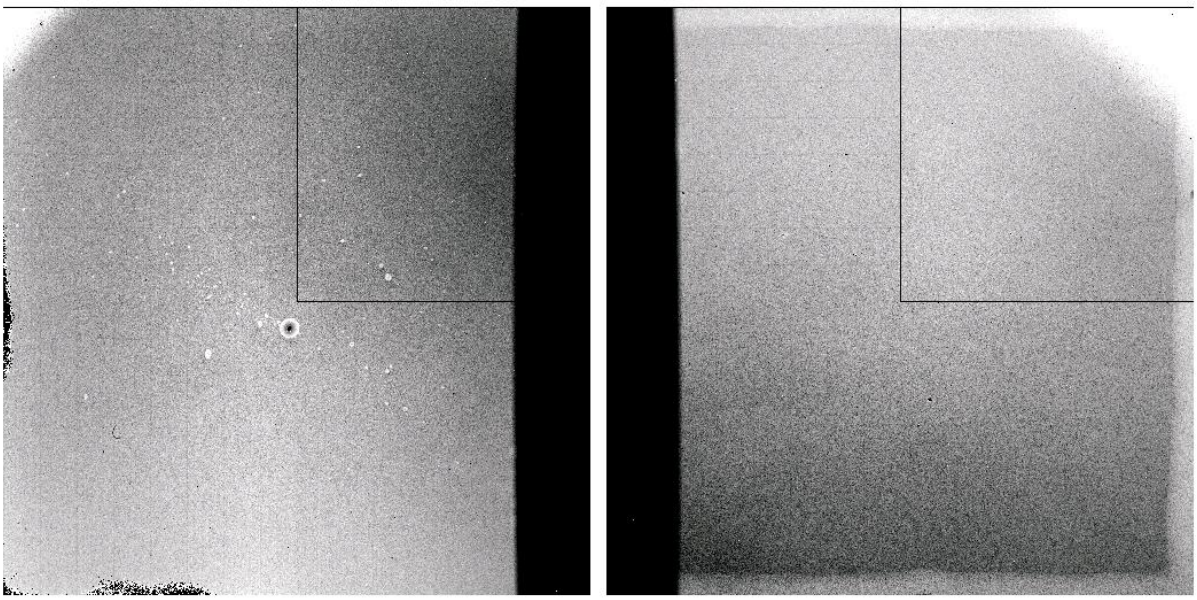


Figure 18: An example of a sky flat in the K-band (left: channel-1, right: channel-2).

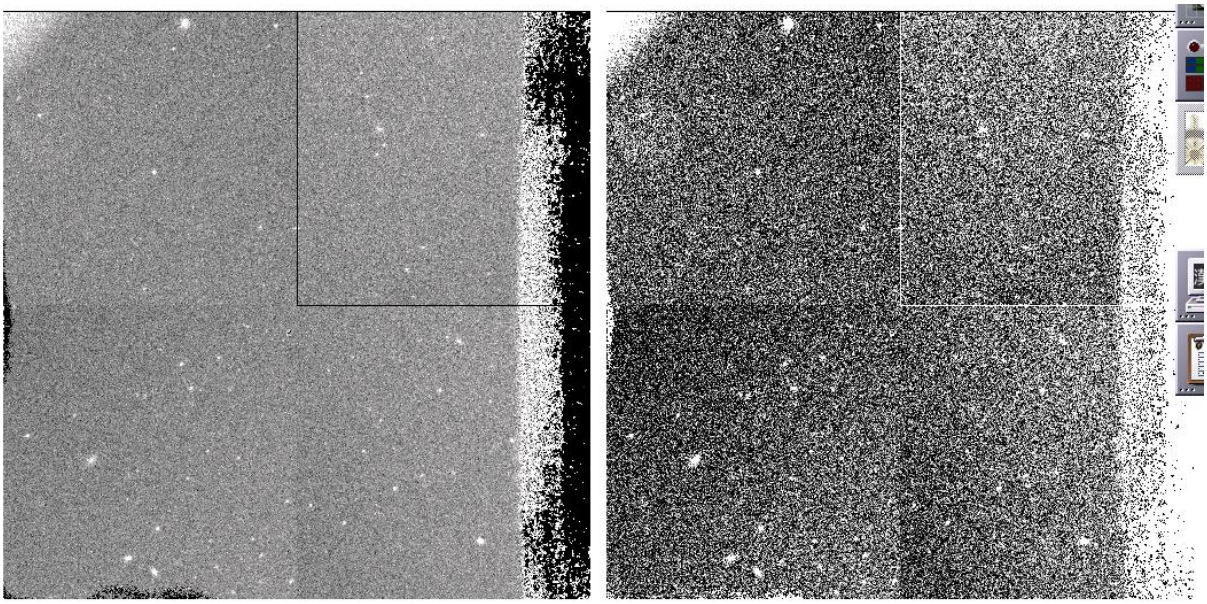


Figure 19: The right hand panel shows an image where flat-fielding was done (`flMCSA00014021.fits`), whereas the left hand panel shows that median-sky was applied (`sbflMCSA14021.fits`).

   (c) Calculating pixel-to-pixel statistics of counts after masking with a median filter.

   (d) Making a mask frame around pixels whose counts are significantly high.

3. Make a self-flat frame:

   (a) Adjusting count levels so that median values of each frame to be combined are identical;

   (b) Median-combining images by applying above object-mask;

   (c) Normalizing the combined image by dividing its median;

4. Flat-fielding, if necessary, using the flat frame made in the process above

For creating a dome flat, you can use the task `mkdome`.

A.2 `sbselfsky`

`sbselfsky` is a task to perform median-sky-subtraction; this task subtracts the sky-background level from each frame given in an input list file using median-sky which will be created from the adjacent frames you specified with 2×`nsf`. This task will help to remove the fringe pattern and/or the systematic patterns attributed to the intrinsic properties of the detectors.

If you increase the numbers of images to make a median-sky flat, you can reduce noise in pixel-to-pixel statistics. On the other hand, the resultant median-sky frame would be degraded by time-variablity of the sky background level over the integration time. Therefore, users have to find an appropriate number of frames to be used by considering the effects of

atmospheric conditions. If your data have been taken under excellent sky conditions, a better result may be obtained by skipping this process.

If your input file of frame list contains three frames, set `nsf=1`. In this case, compare two frames, which are not specified, to find an image that has small count values, for selecting a sky-frame. If you specify `nsf=0`, the task subtracts median-sky, which is made using all the frames, except for the frame from which median-sky is subtracted. Finally, if you set `nsf=-1`, the task subtracts median-sky created from all the images, including the very frame from which median-sky is subtracted.

The standard procedure is:

1. Make a list file for each frame. It should contain names of the frames taken immediately before and after the frame. If the total numbers of the frames listed is smaller than the numbers of frames used for making a median-sky (i.e., `2× nsf`), make a median-sky using all the frames except for the one to be subtracted.
2. Make a median-sky — This task normalizes any frames used for making median-sky if their integration time(s) differ from that of the scientific frames; such a normalization will be done based on the ratio of integration time. Subsequently, this task combines normalized images by applying the object mask.
3. Subtract the sky frame from the scientific frames.

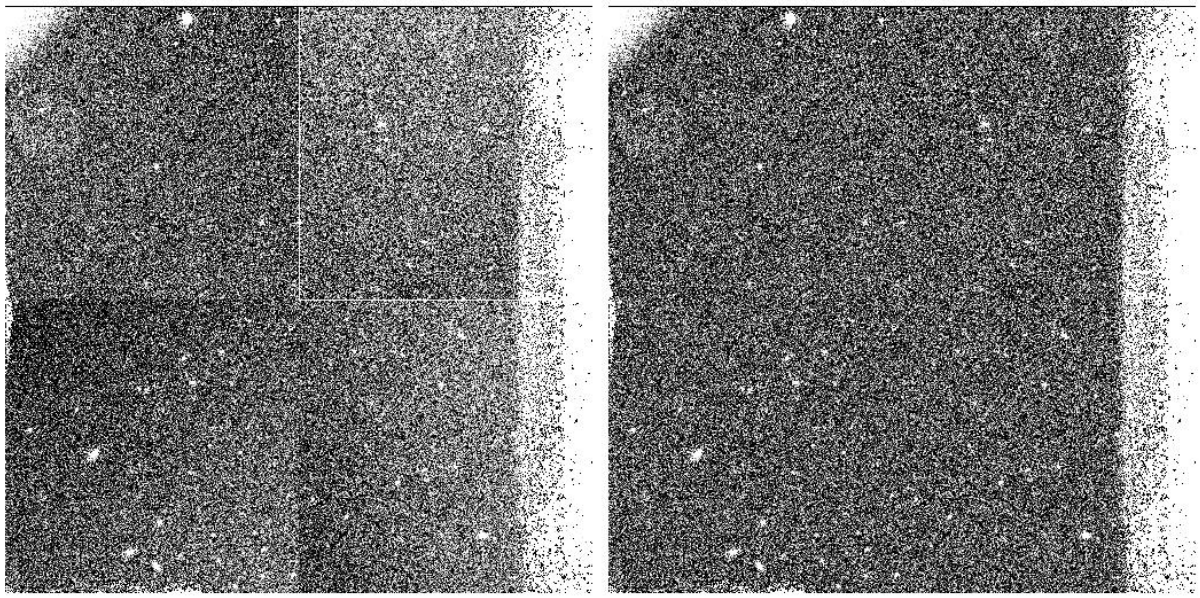


Figure 20: The left panel presents a median-sky-subtracted image (sbflMCSA00014021.fits). The right panel shows an image that was obtained by removing the residual sky emission with the task `qmsepskysb`, then filling gap pixels between the four quadrants with task `qmsepskysb`.

A.3 qmsepskysb

qmsepskysb is a task to fit sky background levels in each quadrant, and subtract them from each quadrant of the original image. In mcsall, this task is used to remove residual sky emission in median-sky-subtracted images.

As mentioned before, MOIRCS reads out the four quadrants separately (but simultaneously) over 11.5 seconds. If the sky background level has changed during this time, e.g., due to passage of clouds, time-variation of the sky-background emission will be recorded in the image(s), causing such a pinwheel pattern as shown in Figure 19. Task qmsepskysb is designed to fit and subtract sky emission in each quadrant, independently (Figure 20).

If there is (are) extended emission (e.g., galaxies) and/or bright star(s) close to the boundaries of the quadrants, qmsepskysb may fail in fitting residual sky-emission, In this case, qmsepskysb may produce artificial gap(s) or add a ripple pattern (Figure 21). We therefore strongly suggest inspecting the output images created by this task. If you have such artifacts, consider removing them by using another task, tiltskycor.

The standard procedure is:

1. Masking bad regions that affect the fitting of sky background emission, this includes
   (a) Masking bad pixels and objects
   (b) Eliminating the right (left) -hand edge of channel-1 (2) data on which photons did not incident (the dark portion in Figure 19) by extrapolating from inner pixels.
   (c) Eliminating pixels having extreme values by applying a median filter of 3×3 pixels (i.e., by smoothing every 3×3 pixels). In addition, one has to eliminate pixels from the upper-left and upper-right corners of channels 1 and 2, respectively, due to vignetting.
2. Extracting each quadrant to fit them using imsurfut
3. Subtracting the fitting fuction from the original images
4. Combining the four quadrant images into an image

A.4 quadcor

The quadcor task fills the gaps between the four quadrants by interpolating from adjacent pixels. This task was originally developed to correct for the image shifts between quadrants, that are caused by errors of pixel storage. This task must be applied to all the data taken before the error was fixed in June 2005. All the data taken as of July 2005 do not have such storage errors but a gap remains; these gaps are filled by blanks, which can be filled with the task fixpix. In practice, these gaps filled by the task may be masked for many scientific frames that will be combined. Hence, you do not need to use the task for many cases. On the other hand, if you work on frames one by one, e.g., analyzing standard stars data, you may have to use this task.

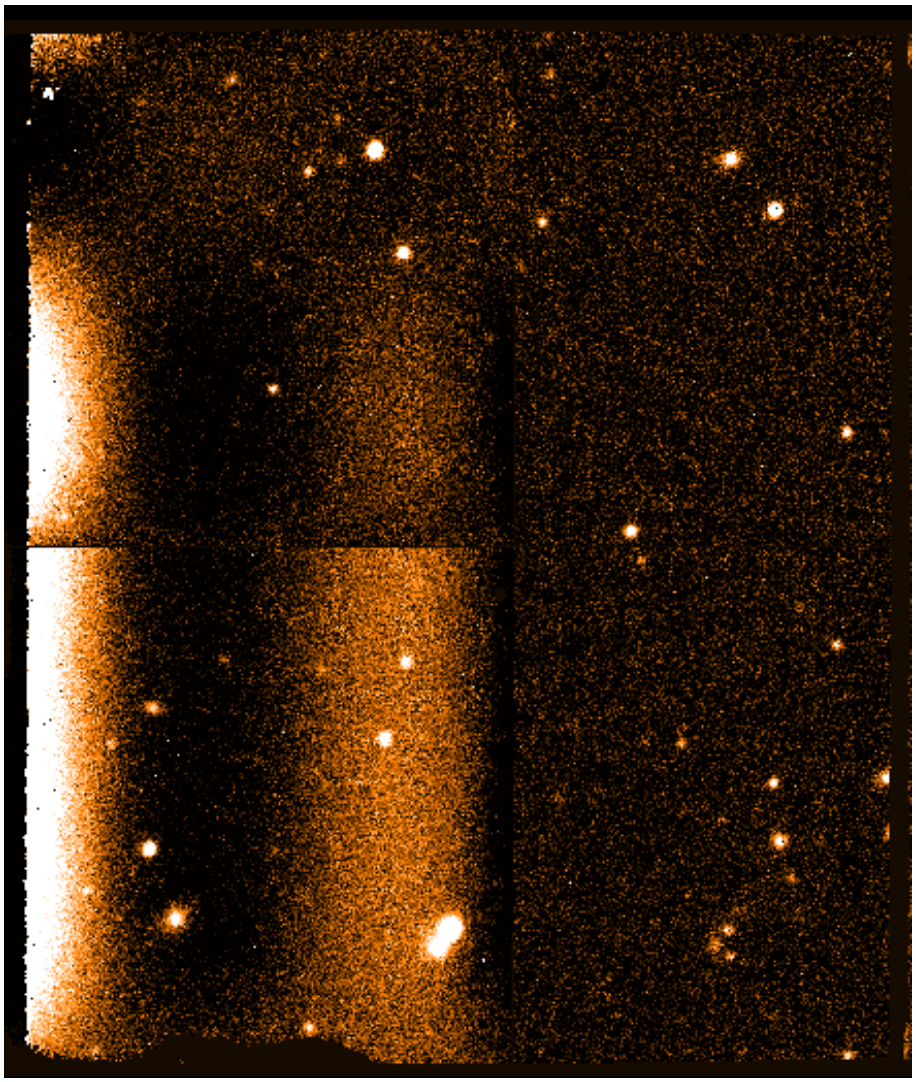


Figure 21: An example of failure in fitting and subtracting sky emission. In this particular case, the second and third quadrants show a systematic pattern because a rail for the slit was left in the FOV.

A.5 `mcsgeocorr`

The task `mcsgeocorr` corrects for distortion of the MOIRCS optics. This task transforms input images into distortion-free images by utilizing the IRAF task `geotran` on the basis of a given distortion parameter file that ends with the extension `.cfg` (described on page 18). Moreover, `mcsgeocorr` corrects for the difference of pixel sizes between the dual channels by forcing them to be $0\farcs117$ pixel^{-1}.

Since the internal optics of MOIRCS may move slightly after each thermal cycle, one has to use the appropriate configuration file, as described in §3.2.7. The observatory staff carries out calibration observations to measure distortion parameters, and makes such a configuration file whenever MOIRCS is thermal-cycled. These configuration files are provided to users through the MOIRCS web page. **Make sure that you select the appropriate configuration file for the period when the data were taken.**

Before correcting distortion, you can add a step to remove pixels where cosmic rays hit using a task `cosmicray`, if necessary. Keep in mind that the presence of a pixel where a cosmic ray hit may affect surrounding pixels because `mcsgeocorr` regrids the original image(s) when they are transformed. Because of this, we strongly suggest clipping such pixels before correcting distortion.

A.6 gsextcat

`gsextcat` is a task to detect objects and make a catalog in preparation for shift-and-add. Since this task utilizes the widely used `SExtractor` package, one has to supply parameters transferred to `SExtractor` accordingly; these parameters are mostly relevant to detection thresholds. Most likely, you should be able to find appropriate parameters by considering the seeing sizes and the amount of cosmic ray hitting. Moreover, our experience suggests that a better result may be obtained by giving a large value of `thresh` in `gsextcat` than the typical value of `DETECT_THRESH` used in `SExtractor`. This is because, after applying distortion correction (i.e., spatially smoothing pixel information), the S/N ratio measured in the corrected image(s) is generally higher than those before the correction. This task will catalogue objects satisfying the given detection threshold, and can reject any doubtful identifications such as objects blending with the other(s) or/and saturated pixels. If you use the `nlimit` option, `gsextcat` makes a catalog for the 69 brightest sources.

The standard procedure is:

1. Extracting a subimage that contains a region specified by the given parameters, then removing bad pixels and filling them by interpolation from surrounding images using `fixpix`
2. Executing `SExtractor`
3. Excluding any objects more extended than the given threshold FWHM from the produced catalog
4. Assessing the intensity-weighted mean of each object using task `center`

A.7 gmkgtrimages

`gmkgtrimages` is a task to shift-and-add for given images using the catalog created by `gsextcat`; this task performs "catalog-matching" to calculate relative positions for each object detected in each frame.

We recommend taking a conservative approach when you carry out "catalog-matching" with the IRAF task `xyxymatch` by setting `fstop = yes`. This allows you to verify results at each step. We also strongly suggest not trying the whole process during your first attempt, as this process tends to fail before completing. If the task fails, it may help you to change parameters such as `nmat`, `tol`, `rat`, and `nrej` in task `xyxymatch`. Once you have successfully found a set of parameters that allow the task to complete catalog-matching, execute the task

once more with `fstop = no` and `fsresume = yes` to the end.

If you still cannot find any reasons for the failure, check the numbers of the sources in each catalog. The numbers of cataloged sources may have changed significantly over a night due to changes in weather conditions, such as variations in the seeing. Removing extreme frames may be the most straightforward way to avoid such a problem. However, if you really need to use such degraded images, you may try to go back to `gsextcat` to expand the detection threshold.

`gmkgtrimages` tries to compile final image by eliminating all the images where `xyxymatch` has failed. You can find a list of these failure images in the ASCII file `failed_cat_`"*your input list*".

`chkbox` is an option to calculate expected relative positions from the FITS headers and to decide the dimension of the output image. Here, the option assumes that all the input frames were made only with parallel translation and NOT by rotation of images. If you want to shift-and-add any images, including rotated one(s), don't use the `chkbox` option, and give `xbox, ybox, xcl,` and `ycl` parameters manually to define a center-coordinate and the dimension of the output image.

After completing catalog-matching, the task will shift all the input images with respect to the first image in the given list. Here, users can select either parallel-translating or rotating images (don't forget to switch off the `chkbox` option). However, it may be worth trying `fitgeo = general` if you want to combine multiple images taken over more than one day or/and those heavily aberrated by the terrestrial atmosphere.

It should be noted that integration times of all the input images are normalized to EXPTIME = 1 second, hence the final combined image has also an integration time of 1 second. The information of the original integration time(s) will be stored in `ORG_EXP` keyword in header of the newly created file.

The task will create not only shifted individual frames before adding (`GTgcSBsbflMCSA000?????.fits`) but also masking frames for each of them (`GTgcSBsbflMCSA000?????.pl`). These files will be useful if you may want to edit these individual mask-frames before combing the final image. For your information, the `skip = yes` option allows you to skip the shifting portion of the task and to perform only the combining one; this requires a list file of shifted images (`GTR_***.lst`).

The standard procedure is:

1. Shifting images
   (a) Executing task `xyxymatch` with object catalogs for each frame to perform catalog-matching with respect to that of the first frame
   (b) Calculating relative positions of the images using `geomap` to obtain conversion equations

(c) Shifting each frame to match with the first frame using the task `geotran`

(d) Transforming the bad-pixel masks as well, as done for the scientific images, and producing mask images in the transformed coordinate.

2. Adding (Combining) Images

(a) Normalizing integration times of the shifted images (if necessary)

(b) Calculating RMS noise levels for each image over an area of [700:1300,700:1300]; the calculated noise levels will be used as a weighting function when combining.

(c) Masking and combining with `imcombine`

B Files created by `mcsall`

0) Making an input list file (`listprep`)
    List file of each channel (``froot''_[12].lst)

1) Masking objects and making self-flat (`mcs_mksflat`)
    List of raw data (or dark-subtracted one) (`bis''inlist''`)
    Object mask (`omskMCSA000?????.pl`) Self-flat frame (specified by `flat`)

2) Flat-fielding (`imarith`)
    Flat-fielded frames (`flMCSA000?????.fits`)
    List file of the flat-fielded frames (`flbs''inlist''`)

3) Median-sky subtraction (`sbselfsky`)
    Median-sky-subtracted frames (`sbflMCSA000?????.fits`)
    List file of the median-sky-subtracted frames (`sbflbs''inlist''`)

4) Subtracting residual sky emission by fitting with the low-order function (`qmsepskysb`)
    Frames from which a low-order fitting is subtracted (`SBsbflMCSA000?????.fits`
    Its list (`SBsbflbs''inlist''`)

5) Correcting gaps between quadrants (`quadcor`)
    Frames whose gaps are processed (`SBsbflMCSA000?????.fits`; the original names remain)

6) Distortion correction (`mcsgeocorr`)
    Distortion corrected frames (`gcSBsbflMCSA000?????.fits`)
    Its list (`dSBsbflbs''inlist''`)

7) Detecting objects (`gsextcat`)
    Catalogs of detected objects (`gcSBsbflMCSA000?????.cat`)
    Its list (`CATdSBsbflbs''inlist''`)

8) Shift-and-add (`gmkgtrimages`)
    The first frame of input list file and matched catalog (`gcSBsbflMCSA000?????.gmp`)
    Database for position-shifting (`gcSBsbflMCSA000?????.gtr` created by `geomap`)
    Position-shifted object frames (`GTgcSBsbflMCSA000?????.fits`)
    Its list (`GTR_''inlist''`)
    Position-shifted mask frames (`GTgcSBsbflMCSA000?????.pl`)
    Combined image (specificied by user)
    Exposure map (`exp_''output''.pl`)
    "Sigma-image" (`sgm_''outout''`)
    Logging files of position-shifting (`gmpdSBsbflbs''inlist''`)
    List of frames that were failed in position-shifting (`failed_cat_''inlist''`)