すばる主焦点カメラ Suprime-Cam 解析用ソフト SDFRED2 マニュアル

ver.2.0.5

SDFRED 開発チーム

2011 年 1 月 14 日
2011 年 12 月 12 日改訂（FN and SDFRED support team）
2013 年 4 月 4 日改訂（FN）
2014 年 9 月 13 日改訂（TT）

このマニュアルは、大内正己さんが作成した、

SDFRED1 マニュアルを基に作成されました。

・はじめに

すばる主焦点カメラ(Suprime-Cam)は、2048pix×4177pix の CCD 10 枚からなるモザイクＣＣＤカメラです。Suprime-Cam で得られた画像データを整約する為のパッケージソフトウェア が Suprime-cam Deep Field REDuction (SDFRED) パッケージです。SDFRED は、Suprime-Cam の膨大な raw データを、できるだけ標準的なパラメータを用いて半自動もしくは自動で処理し、整約済み画像を作ります。この整約済み画像は、そのままサイエンスに用いることができます。

SDFRED はその名の通り、deep field（つまり blank field）のデータ解析を目指して作成されました。Suprime-Cam の視野全体に渡るような巨大な天体が写っているデータや、天体がひどく込み合ったデータに対しては、一部の処理においてうまく動作しない場合があります。また、測光を目的としたソフトウェアですので、天体の精密な形を要求する weak lensing の解析などに使えるかどうかのテストは行われていません。このような場合のデータ解析には、ご自分で出力結果を確認して SDFRED を使用するか、SDFRED 以外のソフトウェアを使用するか、検討することをおすすめします。問題が起こる可能性のある処理に関しては注意書きを載せておきます。

2008 年 7 月に Suprime-Cam は CCD の交換を行いました。本マニュアルでは、この新 CCD（FDCCD）で取得されたデータに対応した、SDFRED 2 について説明します。2008 年 6 月より前に取得されたデータの解析には SDFRED 1 をご利用ください。

・SDFRED 2 の著作権

SDFRED 2 の著作権は、作成した大内正己、古澤久徳、仲田史明及び国立天文台にあります。SDFRED 2 は自由に複製、配布して構いませんが、これを用いた研究の論文には必ず以下の論文を引用することで、著作者を明記してください。

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

・SDFRED 2 の入手方法

SDFRED 2 のパッケージおよび使用マニュアル（本マニュアル）は、

http://www.naoj.org/Observing/Instruments/SCam/sdfred/

から SDFRED 2 の以下のページに行って、取得してください。

http://ww.naoj.org/Observing/Instruments/SCam/sdfred/sdfred2.html.ja

また、今後新しいバージョンの SDFRED２が作られた場合など、SDFRED２に関する情報は全てこのホームページにアップされます。

・SDFRED２以外に必要なソフト

- SExtractor

http://www.astromatic.net/software/sextractor

（Bertin & Arnouts 1996, A&AS, 117, 393）

（参考）開発者は version2.8.6 にて動作確認を行っています。

-IRAF （必ずしも必要はないがあると便利）

入手先： http://iraf.nao.ac.jp/ or http://iraf.net

（参考）開発者は version 2.14.1 にて動作確認を行っています。

-その他の基本ソフト

C コンパイラ、perl、sh、csh

・SDFRED２に必要な環境

メモリ 256MB 程度を持った UNIX 系計算機を使います。ハードディスクの容量は解析するデータの量によりますが、普通は数ＧＢ－数百ＧＢ必要になります。（下記の練習データをつかって解析する場合は、２０ＧＢあれば十分です。）

（参考）開発者は Linux の以下の環境で動作確認を行なっています。

CPU type: x86_64

kernel: version 2.6.18-194.17.1.el5

glibc: version 2.5

gcc: version 4.1.2

・SDFRED２のインストール方法

１、パッケージの展開

ダウンロードした sdfred20130924_mf2.tar.gz を、適当なディレクトリに展開します。（パッケージは配布ウェブにて最新版を確認して下さい）

例）

$ tar xzvf sdfred20130924_mf2.tar.gz

または

$ gunzip –c sdfred20130924_mf2.tar.gz | tar xv

（以下の説明でコマンド例の行頭の%または$はシェルのプロンプトを表しています。お使いのシェルの設定に応じて適宜読み替えてください。）

current directory に sdfred20130924_mf2/ などという名前がついたディレクトリができます。ここで、20130924 という数字は SDFRED２のバージョンで、作成された年、月、日に対応します。（バージョン毎にこの数字は変わります。）

２、コンパイル

展開した SDFRED２をコンパイルします。sdfred20130924_mf2/ というディレクトリに移り configure と make all というコマンドを実行します。

例）

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

これによって、ディレクトリ sdfred20130924_mf2/ の下の bin というサブディレクトリの中にプログラムがインストールされます。

３、path を通す

プログラムがインストールされた sdfred20130924_mf2/bin/ に path を通します。

bash をお使いの場合：

例）

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

などとして~/.bashrc ファイルおよび~/.bash_profile を編集します。

```
PATH=/wa03a/subaru20/sdfred20130924_mf2/bin:$PATH
export PATH
```

などとして、PATH 変数に SDFRED２の実行ファイルディレクトリを加えます。ここで、 /wa03a/subaru20/sdfred20130924_mf2/bin というのは、sdfred20130924_mf2 を置いてあるディレクトリ名なので、ユーザごとに変えてください。なお .bashrc はターミナル起動時、.bash_profile はログイン時に読み込まれるため、両方に反映が必要です。

ファイルを保存終了した後、以下のコマンドを実行し、現在の環境に path 変更を反映します。

$ source ~/.bashrc

csh/tcsh をお使いの場合：

例）

% emacs ~/.cshrc

などとして、

set path=(/wa03a/subaru20/sdfred20130924_mf2/bin $path)

のように set path の中に SDFRED２の bin のディレクトリを指定します。ここで、/wa03a/subaru20/sdfred20130924_mf2/bin というのは、sdfred20130924_mf2 を置いてあるディレクトリ名なので、ユーザごとに変えてください。その後、以下の２つのコマンドを実行して、path の変更を完了してください。

% source ~/.cshrc

% rehash

４、環境変数を設定する

SDFRED２に含まれるシェルスクリプト、Perl スクリプトが正しく動作するように、環境変数 LANG および LC_ALL を以下のように設定します。

bash をお使いの場合：

$ emacs ~/.bashrc （上記３と同様に.bashrc と.bash_profile を編集します。）

export LANG=C （この 2 行を加えます。）

export LC_ALL=C

$ source ~/.bashrc（ファイルを読み込んで環境変数の設定を反映します。）

csh/tcsh をお使いの場合：

% emacs ~/.cshrc （上記３と同様に.cshrc を編集します。）

setenv LANG C （この 2 行を加えます。）

setenv LC_ALL C

% source ~/.cshrc （ファイルを読み込んで環境変数の設定を反映します。）

５．その他の設定

IRAF を使う場合には、カレントディレクトリで、mkiraf を行っておく必要があります。その他、各自の解析環境で必要な設定をあらかじめ完了しておいてください。

一通り設定が完了したら、

% printenv

% which SDFRED２のコマンド名

例）% which namechange.csh

として解析を行おうとしている環境で環境変数や PATH が正しく設定されているかを確認します。

以上で、準備は終了です。ここまでの作業は、解析を行なう環境で最初に１度だけ行なえば良い作業です

・データの準備と整理

作業を開始する前にデータファイルをディレクトリごとに分けておきます。

SDFRED２を用いる際には、各段階で処理される入力ファイルは、カレントディレクトリに存在する必要があります。カレントディレクトリ以外の場所にあるファイルに対して処理を行いたい場合には、それらのファイルにシンボリックリンクを張るなどしてカレントディレクトリ上のファイルとしてアクセス出来るようにしなければなりません。

１．解析練習のための Suprime-Cam データ

国立天文台では、解析練習用の Suprime-Cam データを用意しています。本マニュアルにある手順に従えば、この練習用データから最終画像、および測光用の標準星画像を作れることが確認されています。 SDFRED２を初めて使う方は、実際に自分のデータを解析する前に、この練習用データを用いる事で使用方法や解析の流れを理解することができます。

このデータ解析練習用の Suprime-Cam データは

http://www.naoj.org/Observing/Instruments/SCam/sdfred/data/spcam_training_data_fdccd_1.tar.gz （590MB）

http://www.naoj.org/Observing/Instruments/SCam/sdfred/data/spcam_training_data_fdccd_2.tar.gz （570MB）

から取得できます。必ず両方ダウンロードしてください。

練習用データの展開：

例）

$ tar xzvf spcam_training_data_fdccd_1.tar.gz

$ tar xzvf spcam_training_data_fdccd_2.tar.gz

または

$ gunzip –c spcam_training_data_fdccd_1.tar.gz | tar xv

$ gunzip –c spcam_training_data_fdccd_2.tar.gz | tar xv

この作業を行ったあと、画像の確認を行ってください。どれがオブジェクトフレームか？ どれがフラットに使えるか？標準星として使えるフレームはどれか？さらに ds9 などを使って、星像が歪んでいる画像はないかなどを見てください。この時、観測ログがあればそれも参考にしてください。

練習用データなら、SUPA010998*.fits, SUPA010999*.fits がオブジェクトフレーム（ターゲットの天体を写した画像）、SUPA0109971*.fits が標準星のフレームです。 また、SUPA01102*.fits は、フラットを作るためのフレームです。

以下、本マニュアルでは、それぞれを spcam_training_data_fdccd ディレクトリのあるディレクトリと並べて object/ standard/ flat/というディレクトリに置き、作業を進めることにします。

練習用データでのシンボリックリンク作成の例；

例）

$ mkdir object standard flat

ls の結果は以下のようになるはずです。

$ ls -1

flat

object

spcam_training_data_fdccd

spcam_training_data_fdccd_1.tar.gz

spcam_training_data_fdccd_2.tar.gz

standard

ls -1 の-1 は「マイナス　イチ」です。この後にもある ls -1 の-1 は全て「マイナス　イチ」なので注意。

天体データを object ディレクトリにリンクします

$ cd object/

$ ln -s ../spcam_training_data_fdccd/SUPA010998*.fits .

$ ln -s ../spcam_training_data_fdccd/SUPA010999*.fits .

ブランクマップデータをコピーします

$ cp ../spcam_training_data_fdccd/blankmap* .

$ cp ../spcam_training_data_fdccd/lblank.txt .

標準星データを standard ディレクトリにリンクします

$ cd ../standard/

$ ln -s ../spcam_training_data_fdccd/SUPA0109971*.fits .

フラットデータを flat ディレクトリにリンクします

$ cd ../flat/

$ ln -s ../spcam_training_data_fdccd/SUPA011002*.fits .

standard ディレクトリに 10 FITS ファイル、flat ディレクトリに 30 FITS ファイル、object ディレクトリに 50 FITS ファイルと 2 つの blankmap がある事を確認します。

２．SMOKA からの取得

すばるで得られたデータは１年半後に公開アーカイブ SMOKA で公開されます。

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

ここから取得したデータを使って解析する事も出来ます。この時、フィルターや種別(object/standard など)が、混じっていないことを確認します。

本マニュアルでは、ここからの処理はフラットフレームの作成、ターゲット天体の解析、標準星天体の解析を別々のディレクトリで行ないます。

・SDFRED２を使った画像処理の方法（具体例を含む）

SDFRED２を使って Suprime-Cam の raw データから整約済み画像を作る手順を示します。ここでは２種の手順を分けて説明します。一つはターゲット天体、もう一つは標準星天体です。ターゲット天体の解析は、以下の 12 の処理からなります。

■■解析の流れ■■

処理の内容	(所要時間*)： 用いるプログラム名
（１）画像ファイル名の変換および画像の確認	(1 秒)： namechange.csh
（２）bias 引きおよび overscan の切り取り	(50 秒)： overscansub.csh
（３）flat 作り	(9 分)： mask_mkflat_HA.csh
（４）感度補正 (flat fielding)	(30 秒)： ffield.csh
（５）歪補正(distortion correction)および微分大気差補正	(90 秒)： distcorr.csh
（６）PSF 測定	(100 秒)： fwhmpsf_batch.csh

（７）PSF 合わせ　　(20 分)： psfmatch_batch.csh
（８）sky の差し引き　　(80 秒)： skysb.csh
（９）AG probe の影を自動でマスク　　(30 秒)： mask_AGX.csh
（１０）画像を目で見て、悪い部分をマスク　　(30 秒)： blank.csh など
（１１）組み合わせ規則作り(matching)　　(100 秒)： makemos.csh
（１２）組み合わせ(mosaicing)　　(4 分)： imcio2a

（所要時間）は国立天文台の ana.adc (Intel Xeon 3.0GHz) のローカル作業領域 /mfs を使って、練習用データ（5shot）を解析した場合にかかった時間。この時間は目安で、データ量や使用するコンピュータに依存します。

■■■■■■■■

一方、標準星天体の解析は上記ターゲット天体の処理のうち、一部だけ取り出した以下の４段階と独自の１段階になります。

（S１）画像ファイル名の変換および画像の確認　　： namechange.csh
（S２）bias 引きおよび overscan の切り取り　　： overscansub.csh
（S３）感度補正（flatfielding)　　： ffield.csh
（S４）歪補正（distortion correction)　　： distcorr.csh
（S５）チップ間相対感度補正（mosaicking correction)　　： （特になし）

■■■■■■■■

基本的な流れとしては、ユーザーが各処理に用いるデータファイル名を書いたリストファイルを作り、そのリストファイルを SDFRED２のソフトに与えて実行します。これを上記の１２種の処理に対して行えば整約済み画像（最終画像）が得られます。

ちなみに、処理を終えた画像ファイルには、１つの処理が終わるごとに名前が変わります。最初の処理を除いて、ファイルの名前には処理を代表する頭文字がつけられていきます。以下、SDFRED２で作られる画像ファイル名の一例です。

==========

処理前(raw データ)　SUPA01099886.fits
（１）の処理後　　H090523object038_chihiro.fits
（２）の処理後　　To_RH090523object038_chihiro.fits
（３）の処理後　　（変化なし：flat 画像*mflat*.fits が作られる）
（４）の処理後　　fTo_RH090523object038_chihiro.fits
（５）の処理後　　gfTo_RH090523object038_chihiro.fits
（６）の処理後　　（変化なし：psf サイズの測定結果ファイルが作られる）
（７）の処理後　　pgfTo_RH090523object038_chihiro.fits

（８）の処理後　　spgfTo_RH090523object038_chihiro.fits
（９）の処理後　　AspgfTo_RH090523object038_chihiro.fits
（１０）の処理後　bAspgfTo_RH090523object038_chihiro.fits
（１１）の処理後　（変化なし：組み合わせ規則を示したファイル*.mos が作られる）
（１２）の処理後　（変化なし：ばらばらの画像が一枚に合わされた最終画像ができる）

==========

（注意１）普通、ある処理を行うときは、その直前の処理で作られた画像を入力ファイルとして使います。たとえば、（５）の処理を行うときは、 fTo_RH*.fits と名前のついたファイルです。したがって、ディスクスペースを節約するため、すでに処理に使われなくなった画像ファイルを消してもかまいません。ただし、いつか解析をやりなおす時のために、「（１）の処理後」、「（４）の処理後」、「（９）の処理後」の画像（つまり、 H*.fits, fTo_RH*.fits, AspgfTo_RH*.fits の画像）および *.mos は消さないでおくと便利でしょう。

（注意２）各段階で処理される入力ファイルは、カレントディレクトリに存在しなければなりません。カレントディレクトリ以外の場所にあるファイルに対して処理を行いたい場合には、それらのファイルにシンボリックリンクを張るなどしてカレントディレクトリ上のファイルとしてアクセス出来るようにしなければなりません。
（シンボリックリンク作成の例）
% ln -s [path to the data directory]/*.fits .

・ターゲット天体の処理
上記の（１）から（１２）までの処理について一つ一つ解析方法を解説します。

（１）画像ファイル名の変換および画像の確認

どのファイルがどの日に撮られ、どの CCD によるデータかがすぐわかるように、ファイル名を SUPA.... から H[年月日][file の種類と番号]_[チップ名].fits に変えます。namechange.csh というコマンドを使います。
（注意）この年月日はハワイ時間です。FITS ヘッダに書かれた DATE-OBS（UT）とは１日ずれているので要注意。

コマンド：
% namechange.csh [SUP file names]
ここで

・[SUP file names]： 名前を変えたいファイル名のリスト

上の注意で述べたように、コマンドは、処理に使うファイルの置かれているディレクトリで実行しなければなりません。これは以下の全段階に共通のルールです。

例）

% cd object

天体データのあるディレクトリに移動

% ls -1 SUPA*.fits > namechange.lis

名前を変えるファイル名のリストを作る

% namechange.csh namechange.lis

namechange.csh を実行

ちなみに、

% cat namechange.lis

SUPA01099880.fits

SUPA01099881.fits

SUPA01099882.fits

...

以下の手順で、フラットデータについても名前を変えておきます。

% cd ../flat

% ls -1 SUPA*.fits > namechange.lis

% namechange.csh namechange.lis

実行後）

SUPA...という名前のファイルは以下のようなファイル名に変わる

object ディレクトリ：

H090523object038_chihiro.fits

H090523object038_clarisse.fits

H090523object038_fio.fits

...

flat ディレクトリ：

H090523object077_chihiro.fits

H090523object077_clarisse.fits

H090523object077_fio.fits

...

実行後のチェックポイント）

— SUPA...という名前のファイルがちゃんと H...という名前に変わっているはずです。ls コマンドで確認できます。

（参考）Suprime-Cam の１０枚の CCD の並び方は次のようになっています。

AG プローブの位置

chihiro	clarisse	fio	kiki	nausicaa
ponyo	san	satsuki	sheeta	sophie

（注意）上記の例で、‘% ls -1 SUPA*.fits > namechange.lis’ を実行する際、ls に-F オプションがエイリアス設定されていると、シンボリックリンクとなっている SUPA01099880.fits などの末尾に @ が出てきてしまいます。’cat namechange.lis’ できちんと確認してください。この後も ls -1 のコマンドを使う際には注意が必要です。

（２）bias 引きおよび overscan の切り取り

各画像から bias を引きます。さらに、いらなくなった overscan の部分を切り取ります。overscansub.csh というコマンドを使います。bias 引きは、overscan の値を bias 値と見なして行われます。まず、CCD の左右にある serial overscan のそれぞれの行のメジアンを各行から引きます。その後 CCD の上下にある parallel overscan のそれぞれの列のメジアンを各列から引くことで bias 差し引きが行われます。
以下の操作を行います。

コマンド：

% overscansub.csh　[overscansub.lis]

ここで

・[overscansub.lis]：bias を引きと overscan の部分を切り取りたいファイル名のリスト

例）

% cd ../object

天体データのあるディレクトリへ移動

% ls -1 H090*.fits > overscansub.lis

解析に使う画像のリストを作る

% overscansub.csh overscansub.lis

overscansub.csh を実行

ちなみに、

% cat overscansub.lis

H090523object038_chihiro.fits

H090523object038_clarisse.fits

H090523object038_fio.fits

...

サンプルデータの場合、以下の手順で、フラットデータについても bias を引いておきます。(3) 節の「flat 作り」を、上で解析した object データのみから行う場合など、以下の手順が不要な場合があります。

% cd ../flat

% ls -1 H090*.fits > overscansub.lis

% overscansub.csh overscansub.lis

実行後）

以下のようなファイル(overscan bias 引きと切り取り済み画像）ができる

To_RH090523object038_chihiro.fits

To_RH090523object038_clarisse.fits

To_RH090523object038_fio.fits

...

実行後のチェックポイント）

— 画像上で天体が写っていない部分（背景）のカウント数が、2011 年 7 月までのデータではおよそ 100－300 カウント、それ以降のデータでは 1500 カウント程度、元の画像に比べて減っているはずです。これは、画像から bias を引いた結果です。引かれるカウント数（bias の値）は、CCD ごと、pixel ごとに異なります。

— overscan の部分は切り取られるため、処理後は若干画像が小さくなります。IRAF の imhead などを使って、画像の pixel 数が変化しているのを見てとることができます。（例： cl> imhead H*.fits ）

（3）flat 作り

画像内での感度補正に使う画像(flat)を作ります。mask_mkflat_HA.csh というコマンドを使います。(天体や AG probe の影などは自動でマスクされ、median flat が作られます。)
フラット作りに使える画像は、大きく分けて 3 種類あります。「blank field の画像」、「twilight 画像」、そして「dome 画像」です。最も良質のフラットは、「blank field の画像」を使うと作ることができると考えられています。「blank field の画像」は、解析しようとする天域と同じでも、100-200pix を越えるような大きな天体が無ければ問題ありません。練習用データでは、比較的近い距離にある銀河団（Abell1689）の画像を扱っています。特に銀河団の中心部には明るい天体が数多く存在するため、これらの画像からフラットを作るのは不適切になります。そこで、今回は「dome 画像」からフラットを作ります。
練習用データではフラットは flat ディレクトリ内で作成し、それを後ほど標準星の解析でも利用します。

コマンド：
% mask_mkflat_HA.csh [mkflat.lis] [head name] [lower value] [uppwer value]

ここで
・[mkflat.lis] ： フラットを作るのに使うファイルのリスト
・[head name] ： 作られるフラットの頭の部分の名前
・[lower value]： 作られるフラットの最低値(それ以外はマスク）[0.4 を推奨]
・[uppwer value]： 作られるフラットの最大値(それ以外はマスク）[1.3 を推奨]

例）
% cd ../flat
フラットデータのあるディレクトリへ移動
% ls -1 To_RH090*.fits > mkflat.lis
フラットを作るのに使うファイルのリストを作る。
% mask_mkflat_HA.csh mkflat.lis dome 0.4 1.3
mask_mkflat_HA.csh を実行
ちなみに、
% cat mkflat.lis
To_RH090523object077_chihiro.fits
To_RH090523object077_clarisse.fits
To_RH090523object077_fio.fits
...

実行後）
以下のような名前のファイル（フラット画像）ができる
dome_mflat_chihiro.fits
dome_mflat_clarisse.fits
dome_mflat_fio.fits
...

実行後のチェックポイント）

— フラット画像（dome_mflat*）は、1.0 カウントを中心にカウントがばらついているはずです。また、この画像は場所毎に極端なデコボコは少なく、滑らかにカウントが変わっている場合が多いです。（ただしUバンドや z バンドより赤いバンドの画像はデコボコが多いです。）しかし、デコボコといっても連続的にカウント数が減っているはずです。もし、カウントの場所ごとの変化が不連続になるようでしたら、うまくフラット画像が作れていない可能性があります。
ちなみに、-32768 という極端に小さいカウントが入っている pixel がみられるはずです。これは、ブランクする pixel に対して埋め込まれる値で、異常な結果ではありません。ただし、この -32768 というカウントのピクセルが、広い面積で発生するような場合は、コマンドのパラメータのうち、[lower_value], [upper_value] が不適切である可能性があります。

（注意１）フラットは、原理的に３ショット分の画像（つまり各ＣＣＤに３枚の画像）があれば作れます。しかし、フラット作りに使う画像の枚数が少なければ少ないほど、天体の影響やノイズの影響を受けやすくなります。したがって、少なくとも６－７ショットの画像、理想的には２０ショット以上の画像からフラットを作ることをお勧めします。

（注意２）「blank field の画像」、「twilight 画像」、「dome 画像」のうち２種類もしくは３種類の画像を混ぜてフラットは作らないでください。３種類の画像はそれぞれ背景光の傾きが異なるため、出来上がるフラット画像がおかしくなる場合があります。（たとえば、カウントが不連続な横筋が入るなど）。これは本パッケージのアルゴリズムと関係しています。

（注意３）このコマンドでは、bad column, hot pixel などの位置を、パラメターファイルに基づいて自動的にマスクさせています。

（4）感度補正（flat fielding）

（3）節で作成した flat 画像を用いて、１画像内での相対感度の補正をします。

コマンド：

% ffield.csh [ffield_mf.lis] [ffield_im.lis]

ここで

・[ffield_mf.lis]： 使用するフラット画像のリスト

・[ffield_im.lis]： 感度補正を行いたい画像のリスト

例）

% cd ../object

天体データのあるディレクトリへ移動

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

(3)で作ったフラットを object ディレクトリにリンクします。

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

(3)で作ったフラットの名前が載ったリストを作る

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

感度補正を行う画像のリストを作る

% ffield.csh ffield_mf.lis ffield_im.lis

ffield.csh を実行

ちなみに、

% cat ffield_mf.lis

dome_mflat_chihiro.fits

dome_mflat_clarisse.fits

dome_mflat_fio.fits

...

% cat ffield_im.lis

To_RH090523object038_chihiro.fits

To_RH090523object038_clarisse.fits

To_RH090523object038_fio.fits

...

実行後）

以下のような名前のファイル（感度補正済みの画像）ができます

fTo_RH090523object038_chihiro.fits
fTo_RH090523object038_clarisse.fits
fTo_RH090523object038_fio.fits
...

実行後のチェックポイント）

— うまく出来た感度補正済み画像を見ると、背景光のカウント数がほぼ平らになっているはずです。また、視野の端にある CCD つまり、chihiro, nausicaa, ponyo, sophie などにあった光学系によるケラレ（半月状の形）もほとんど無くなっているはずです。しかし、たまにこの背景光のムラが消えず、カウントが数パーセント高く（または低く）なってしまうことがあります。

（５）歪補正(distortion correction)および微分大気差補正

主焦点光学系および微分大気差による画像中の歪みを同時に補正します。

コマンド：
% distcorr.csh　[distcorr.lis]

ここで
・[distcorr.lis]： 歪み補正する画像のリスト

例）
% ls -1 fTo_RH090*.fits > distcorr.lis
歪補正を行うオブジェクト画像のリストを作る
% distcorr.csh distcorr.lis
distcorr.csh を実行
ちなみに、
% cat distcorr.lis
fTo_RH090523object038_chihiro.fits
fTo_RH090523object038_clarisse.fits
fTo_RH090523object038_fio.fits
...

実行後）
以下のような名前のファイル（歪み補正済みの画像）ができます。

gfTo_RH090523object038_chihiro.fits
gfTo_RH090523object038_clarisse.fits
gfTo_RH090523object038_fio.fits
...

実行後のチェックポイント）

— 出来上がった画像は、元の画像と比べると数 pix から数十 pix 歪んでいるように見えます。画像の左右（または上下）の縁の部分に注目すると、その違いが分かりやすいです。この処理は、ヘッダーの位置情報を読んできて、計算式に当てはめて補正量を決めています。したがって、この処理で画像作りに失敗することは（原理的に）ありません。

（参考）なお、本来はこの歪みの量は波長によって異なるのですが、SDFRED 2 で採用している値は、2008 年 6 月まで Suprime-Cam で使用していた MIT CCD で R band の観測を行ったときの量を元にしています。2008 年 7 月以降の新 CCD (FDCCD) では、このパラメータの確認と最適化は行われていませんのでご注意ください。

（6）PSF 測定

最終的に画像を重ね合わせる際には、重ね合わせる画像の PSF が同じでなければなりません。
シーイングによる画像間の PSF の大きさの違いは次節（7）で合わせますが、その前に、この節では画像間の PSF の大きさがどのくらい異なるかを予め測っておき、目標とする FWHM（次節の[target FWHM]）を決めておきます。この作業には、fwhmpsf_batch.csh というプログラムを使います。

コマンド：
% fwhmpsf_batch.csh [fwhmpsf_batch.lis] [max number of objects] [min peak flux] [max peak flux] [min FWHM] [max FWHM]

ここで

・[max number of objects] ： PSF の測定に使う stellar object の数
・[min peak flux] ： PSF 測定に使う stellar object の flux peak の許容最小値
・[max peak flux]： PSF 測定に使う stellar object の flux peak の許容最大値
・[min FWHM]： PSF 測定に使う stellar object の FWHM の許容最小値
・[max FWHM]： PSF 測定に使う stellar object の FWHM の許容最大値
・[fwhmpsf_batch.lis]：PSF を測る画像のリスト

例）

% ls -1 gfTo_RH090*.fits > fwhmpsf_batch.lis

PSF の大きさを測る画像のリストを作る

% fwhmpsf_batch.csh fwhmpsf_batch.lis 50 2000 40000 2.0 7.0

fwhmpsf_batch.csh を実行

ちなみに、

% cat fwhmpsf_batch.lis
gfTo_RH090523object038_chihiro.fits
gfTo_RH090523object038_clarisse.fits
gfTo_RH090523object038_fio.fits
...

実行後）

以下のような出力が得られます。

results

gfTo_RH090523object038_chihiro.fits 4.10 1 5 12 0 0
gfTo_RH090523object038_clarisse.fits 4.00 0 3 16 18 0
gfTo_RH090523object038_fio.fits 4.10 1 11 21 0 0
...
3.5 |****
3.6 |*
3.7 |******
3.8 |**
3.9 |*
4.0 |******

4.1 |*******
4.2 |******
4.3 |*

”gfTo_RH090523object038_chihiro.fits　4.10　1 5 12 0 0” などで表示される出力の前半は、個々の画像に対する PSF の測定結果を示します。各列の意味は、

1 列目: 画像の名前
2 列目: 求められた平均的な PSF の FWHM （mean FWHM; pixel 単位）
3 列目: (mean FWHM – 0.2 pixel) を中心とする 0.1 pixel 以内の FWHM を持つ天体の数
4 列目: (mean FWHM – 0.1 pixel) を中心とする 0.1 pixel 以内の FWHM を持つ天体の数
5 列目: mean FWHM を中心とする 0.1 pixel 以内の FWHM を持つ天体の数
6 列目: (mean FWHM +0.1 pixel) を中心とする 0.1 pixel 以内の FWHM を持つ天体の数
7 列目: (mean FWHM +0.2 pixel) を中心とする 0.1 pixel 以内の FWHM を持つ天体の数

３－７列目の情報は、うまく PSF 合わせが出来たかどうか見る指標になります。うまく出来た画像は、５列目あたりを中心に３列目または７列目に行くにつれ、数が減っていく傾向があるはずです。また、うまく処理が出来た場合、５列目の値（中心の値）はたいてい１０を超えています。このような傾向が見られないものについては、下記で説明する方法により、手動で PSF の大きさを測って確認することをお勧めします。

後半は、測定値 vs.画像の数のヒストグラムになります。例えば、4.1 にある * マークの一つは ”gfTo_RH090523object038_chihiro.fits　4.10　1 5 12 0 0” によるものです。このヒストグラムを基に、次節 (7) PSF 合わせ で使用する target FWHM を決定します。各自の科学的目標に合わせ、慎重に決めてください。例えば、上の結果の場合、「4.3 に全て揃える」「4.3 は外れ値と見做し、 4.2（もしくは 4.1）に揃える」などが考えられます。このヒストグラムは、FWHM が悪い画像を解析に使用するかどうかを判断するための指標にすることも出来ます。

手動で PSF を測るには IRAF の imexam が便利です（ds9 を立ち上げ、 cl> imexam *.fits とし画像上で星らしい天体を探し、r または a を押して FWHM を測る)。1 フレームの中から星らしい天体を選ぶためには、多少の練習が必要です。基本的にはバッドピクセルや宇宙線イベントではない通常の天体の中で、一番小さくかつ真円状に結像しているものを選びます。通常、星でない銀河などの天体はそのフレームの PSF（星らしい天体が持つべきカウントのプロファイル）に比べて有意に広がった形状をしています。観測条件によっては PSF が十分に円くないこともありますが、IRAF の imexam などでいくつかの候補天体

の contour と FWHM を見ていくと、数~10 天体ほど見たところで、そのフレーム内で共通のおおよその PSF が分かるでしょう。いったん PSF が分かれば、後はその PSF と同等のサイズ・形状の天体を探していけばそれが星らしい天体（QSO などの恒星状天体も含みます）ということになります。
この際、一つ注意すべき点があります。天体の FWHM の大きさは、測るソフトウェアにより値が変わります。これは用いるフィッティングの関数の違いなどによるものです。SDFRED２では、SExtractor が測った FWHM の値を採用しています。これは、IRAF で測った FWHM と値が違います。したがって、IRAF などを使って手動で PSF を測って確認する場合は、FWHM の絶対値があっているかどうかを見るのではなく、いくつかの画像を IRAF で測って同じような FWHM に落ち着いているかどうかを確認することになります。

（注意１）fwhmpsf_batch.csh や、次の psfmatch_batch.csh などは、 Suprime-Cam の視野の数分の１から全面に広がる巨大な天体の画像（球状星団や近傍銀河など）に用いると誤作動する可能性があります。これは、混み合っている天域では、stellar object を自動で探すのが難しくなるためです。うまく動作しているかどうかを手動で確かめてください。うまくいっていない場合は、手動で PSF 合わせを行う必要があります。

（注意２）fwhmpsf_batch.csh、および次の節で説明する psfmatch_batch.csh では、画像に載っている宇宙線の影響が大きいと正常に動作しない時があります。このような場合は、宇宙線除去を行う必要があるかもしれません。宇宙線除去を行うソフトとしては、例えば L.A.Cosmic (http://www.astro.yale.edu/dokkum/lacosmic/) があります。SDFRED２では、L.A.Cosmic の IRAF 用のソフトを使用すれば正常に画像解析ができることを確認しています。L.A.Cosmic を使用する際は、フラットフィールド直後（fTo_*.fits）の画像に使用することをお勧めします。

（参考１） fwhmpsf_batch.csh は多数の画像の PSF を一度に測るプログラムですが、一つの画像についてだけ PSF を測りたい場合は fwhmpsf.csh を使ってください。使い方は、fwhmpsf_batch.csh のリスト名を与える部分を画像名にするだけです。

例）
% fwhmpsf.csh gfTo_RH090523object038_chihiro.fits 50 2000 40000 2.0 7.0

実行後）
以下のような出力が得られます。

gfTo_RH090523object038_chihiro.fits　4.10　1　5　12　0　0

この結果、gfTo_RH090523object038_chihiro.fits という画像は、PSF の大きさ(FWHM)が 4.1pix であると分かります。

（参考２）fwhmpsf_batch.csh に使う各種パラメータの決め方
以下の５つのパラメータ

[max number of objects]
[min peak flux]
[max peak flux]
[min FWHM]
[max FWHM]

は、解析する画像によって変える必要があります。これは、画像の質がバンドや積分時間、取得時の天候などによって変わるためです。
これら５つのパラメータは、PSF を測るために使う画像中の星の選ぶ基準になります。ユーザーが選んだ５つのパラメータが適当かどうかを見るためには starselect.csh を使うと便利です。starselect.csh は、ある画像の中から星を選び、選んだ星の位置を示した ds9 の region ファイルを作るプログラムです。

コマンド：
% starselect.csh [image name] [max number of objects] [min peak flux] [max peak flux] [min FWHM] [max FWHM] [output file]

ここで
・[image name]：星の位置を調べたい画像の名前
・[max number of objects]：stellar object の数（psfmatch_batch.csh と同じ）
・[min peak flux] : stellar object の flux peak の許容最小値（psfmatch_batch.csh と同じ）
・[max peak flux] : stellar object の flux peak の許容最大値（psfmatch_batch.csh と同じ）
・[min FWHM]：stellar object の FWHM の許容最小値（psfmatch_batch.csh と同じ）
・[max FWHM]：stellar object の FWHM の許容最大値（psfmatch_batch.csh と同じ）
・[output file]：stellar object の位置を示した ds9 の region ファイル（出力）

例）
% starselect.csh gfTo_RH090523object038_chihiro.fits 50 2000 40000 2.0 7.0

output.reg

starselect.csh を実行

実行後）
以下のような名前のファイル(stellar object の位置を示したファイル)ができる。
output.reg

stellar object の位置は ds9 を使って表示することができます。

% ds9 gfTo_RH090523object038_chihiro.fits

として画像を表示させ、上側にあるボタン region をクリック、その後、 load をクリックして、output.reg　と打ち込みリターン。選ばれた天体は、画像上に緑色の丸で示されます。選ばれた天体の半分以上が本物の stellar object だったら用いた５つのパラメータは妥当と考えて良いです。これと同じ値を psfmatch_batch.csh のパラメータに使えます。
もし、解析する画像の中に質が大きく異なるものが含まれる場合は、この作業を異なる質の画像に対しても行ってください。質の異なる画像でも、決定した５つのパラメータで本当に正しく stellar object が選べるかどうかチェックしてください。全ての画像でうまく動くパラメータを探してください。万一、そういったパラメータの組み合わせが無いようでしたら、 psfmatch_batch.csh は、画像の種類ごと別々に行ってください。

（7）PSF 合わせ

シーイングによる画像間の PSF の大きさの違いを合わせ、（6）で求めた値に揃えます。今回使用するサンプルデータでは、（6）節の結果から、target FWHM を 4.1 にするものとして説明します。

コマンド：
% psfmatch_batch.csh　[psfmatch_batch.lis] [max number of objects] [min peak flux] [max peak flux] [min FWHM] [max FWHM] [target FWHM]

ここで
・[psfmatch_batch.lis]：PSF 合わせをする画像のリスト
・[max number of objects]：(fwhmpsf_batch.csh と同じもの)
・[min peak flux]：(fwhmpsf_batch.csh と同じもの)

・[max peak flux]：(fwhmpsf_batch.csh と同じもの)
・[min FWHM]：(fwhmpsf_batch.csh と同じもの)
・[max FWHM]：(fwhmpsf_batch.csh と同じもの)
・[target FWHM]：目標にする画像の FWHM（元々この値より FWHM が大きい画像は
実行後、単にコピーされる）

例）

% ls -1 gfTo_RH090*.fits > psfmatch_batch.lis

PSF 合わせを行うオブジェクト画像のリストを作る

% psfmatch_batch.csh psfmatch_batch.lis 50 2000 40000 2.0 7.0 4.1

psfmatch_batch.csh を実行

ちなみに、

% cat psfmatch_batch.lis
gfTo_RH090523object038_chihiro.fits
gfTo_RH090523object038_clarisse.fits
gfTo_RH090523object038_fio.fits
...

実行後）

以下のような名前のファイル（PSF 合わせ済みの画像）ができる

pgfTo_RH090523object038_chihiro.fits
pgfTo_RH090523object038_clarisse.fits
pgfTo_RH090523object038_fio.fits
...

実行後のチェックポイント）

― プログラムの終了時に例えば以下のように処理の結果が示されます。

results (after psfmatch)
pgfTo_RH090523object038_chihiro.fits 4.10 0 5 15 0 0
pgfTo_RH090523object038_clarisse.fits 4.10 0 2 16 18 0
pgfTo_RH090523object038_fio.fits 4.10 0 0 16 18 0
...

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

“pgfTo_RH090523object038_chihiro.fits　4.10　0 5 15 0 0” などで表示される出力の前半は、（６）節と同じく個々の画像に対する PSF の測定結果を示します。各列の意味は、

1 列目: 画像の名前
2 列目: 求められた平均的な PSF の FWHM （mean FWHM; pixel 単位）
3 列目: (mean FWHM) – 0.2 pixel を中心とする 0.1 pixel 以内の FWHM を持つ天体の数
4 列目: (mean FWHM) – 0.1 pixel を中心とする 0.1 pixel 以内の FWHM を持つ天体の数
5 列目: mean FWHM を中心とする 0.1 pixel 以内の FWHM を持つ天体の数
6 列目: (mean FWHM) +0.1 pixel を中心とする 0.1 pixel 以内の FWHM を持つ天体の数
7 列目: (mean FWHM) +0.2 pixel を中心とする 0.1 pixel 以内の FWHM を持つ天体の数

後半のヒストグラムも（６）節と同じく、測定値 vs.画像の数を表します。このヒストグラムが target FWHM (今回の場合は 4.1) を中心に分布しているか確認してください。

（８）sky の差し引き

背景光（sky）を画像から差し引きます。

コマンド：
% skysb.csh　[skysb.lis] [sky-mesh]

ここで
・[skysb.lis]：sky を引く画像のリスト
・[sky-mesh]：sky を決める mesh の大きさ（pix 単位, 少なくともターゲット天体の大きさの２倍以上の値でなくてはいけない。）

例）
% ls -1 pgfTo_RH090*.fits > skysb.lis
sky 引きを行うオブジェクト画像のリストを作る

% skysb.csh skysb.lis 64

skysb.csh を実行

ちなみに、

% cat skysb.lis

pgfTo_RH0900523object038_chihiro.fits

pgfTo_RH0900523object038_clarisse.fits

pgfTo_RH0900523object038_fio.fits

...

実行後）

以下のような名前のファイル（sky 引き済み画像）ができる

spgfTo_RH090523object038_chihiro.fits

spgfTo_RH090523object038_clarisse.fits

spgfTo_RH090523object038_fio.fits

...

実行後のチェックポイント）

― 処理済みの画像は、背景（天体の無い場所）が、およそ０カウントを中心に＋/－にばらついているはずです。

（注意）（６）節と同様に、Suprime-Cam の視野の数分の１から全面に広がる巨大な天体の画像に用いると誤動作する可能性があります。これは、混み合っている天域では、背景光の高さを見積もることが難しくなるためです。うまく動作しているかどうかを手動で確かめてください。うまくいっていない場合は、skysb.csh を用いず、手動で背景光の高さを決めて差し引きを行う必要があります。

（９）AG probe の影を自動でマスク

画像上に AG probe によってできた影（上側の CCD に現れる）をマスクします。マスクする範囲は、fits ヘッダーにある AG probe の位置情報から自動的に決められます。

コマンド：

% mask_AGX.csh　[mask_AGX.lis]

ここで

・[mask_AGX.lis]：AG probe の影にマスクをかける画像のリスト

例）

% ls -1 spgfTo_RH090523*.fits > mask_AGX.lis

マスクをかけるオブジェクト画像のリストを作る

% mask_AGX.csh mask_AGX.lis

mask_AGX.csh を実行

ちなみに、

% cat mask_AGX.lis

spgfTo_RH090523object038_chihiro.fits

spgfTo_RH090523object038_clarisse.fits

spgfTo_RH090523object038_fio.fits

...

実行後）

以下のような名前のファイル（AG probe の影をマスクした画像）ができる。

AspgfTo_RH090523object038_chihiro.fits

AspgfTo_RH090523object038_clarisse.fits

AspgfTo_RH090523object038_fio.fits

...

実行後のチェックポイント）

― AG probe の影が映っている画像は、上側が数百 pix かそれ以下の大きさでマスクの値(-32768)が埋め込まれているはずです。AG probe の影は、いつも現れるものではありません。また、現れるにしても AG probe 側にある 5 つの CCD （chihiro, clarisse, fio, kiki, nausicaa）しかありません。AG probe の影が画像に映っていない場合は、何も処理されません。

（１０）画像を目で見て、悪い部分をマスク

観測状況（例えば人工衛星の通過跡や明るい星の周囲のハロー）や装置の不具合（例えば CCD にあるデッドカラム）などで、画像の一部分がおかしくなっている場合があります。このような部分は、マスクして最終画像には残らないようにします。基本的には画像を一

枚一枚目で見て、悪い部分を見つけます。正常な場所では、背景(sky)の値は０を中心にばらついています。sky の値が系統的に０からずれている領域は、おかしい部分の候補となります。おかしい部分は、その形（または用途）などによって以下に述べる３つのプログラム(line_blank, circular_blanks, blank.csh)を用いてマスクします。

（参考）１つのターゲット天体に対して観測データが複数ショットある場合、仮におかしい部分をマスクするのを忘れてしまっても最終画像がおかしくなるケースはあまりありません。普通、マスクかけは細かいところまで行う必要はありません。

（I）satellite trail をマスクする方法（一枚ずつ処理）： line_blank
露出している間に視野中を人工衛星が通ったためにできた跡を satellite trail と呼びます。観測する方向によっては、satellite trail が数ショットに一枚程度現れることがあります。

コマンド：
% line_blank　[input image]　[x1] [y1] [x2] [y2] [width]　[blank value] [output image]

ここで
・[input image]：マスクをかける画像名
・[x1] [y1] [x2] [y2]：satellite trail が(x1,y1)と(x2,y2)を通る
・[width]：satellite trail の太さ
・[blank value]：マスク領域に使う値(通常-32768 を使う)
・[output image]：マスクをかけた後の画像名

例）
% cat lblank.txt
line_blank AspgfTo_RH090523object038_kiki.fits 10 3836 1351 8 90 -32768 lAspgfTo_RH090523object038_kiki.fits
…
上記の AspgfTo_RH090523object038_kiki.fits に対する line_blank コマンドでは、中心が　(X,Y)=(10,3836),(1351,8)を通る太さ 90pix のマスクをかけている

% bash < lblank.txt
line_blank を実行

実行後）

以下のような名前のファイル（マスクした画像）ができます。

lAspgfTo_RH090523object038_kiki.fits
lAspgfTo_RH090523object038_sheeta.fits
lAspgfTo_RH090523object038_sophie.fits
…

実行後の画像とマスクする前の画像を見比べて、マスクするべき領域が適切かどうか、チェックしてください。

（II） 円形の領域をマスクする場合（一枚ずつ処理）： circular_blanks

コマンド：

% circular_blanks [input image] [blanklist(x y radius)] [blank value] [output image]

ここで

・[input image]：マスクをかける画像名
・[blanklist(x y radius)]：円形マスクの中心位置(x,y)と半径 radius が
書かれたリスト（複数行書けば複数個の円形マスクが一度にできる）
・[blank value]：マスク領域に使う値(通常-32768 を使う)
・[output image]：マスクをかけた後の画像名

例）

% circular_blanks lAspgfTo_RH090523object038_chihiro.fits blanklist -32768 clAspgfTo_RH090523object038_chihiro.fits

blanklist に書かれた中心位置と半径をもつマスクをかける

ちなみに、

% cat blanklist
356 1835 80
1202 3582 100
...

それぞれ(x,y,r)=(356,1835,80), (1202,3582,100)…の円に対応

実行後）

以下のような名前のファイル（マスクした画像）ができます。

clAspgfTo_RH090523object038_chihiro.fits

上記の clAspgfTo_RH090523object038_chihiro.fits に対するコマンドは、あくまで例であり、実際には円形領域のマスクをする必要はありません。その他の画像についても、今回の練習データでは、円形領域のマスクをするべき画像はないと思われますが、気付いた領域がありましたらマスクしてみてください。

（III）四角形の領域をマスクする場合（バッチ処理ができる）： blank.csh

コマンド：
% blank.csh　[blank.lis]

ここで
・[blank.lis]：マスク（四角形の領域）をかける画像のリスト
これに加え、blank.csh の入力パラメータとして、同じディレクトリに
blankmap_[画像名(.fits はつかない）] という名前の blank 位置を示したファイルを
同じディレクトリに置いておく必要がある。

以下の処理は、練習データに対して行ってください。

例）
% ls -1 AspgfTo_RH090*.fits > blank.lis
% ls -1 lAspgfTo_RH090*.fits >> blank.lis
マスクをかけるオブジェクト画像のリストを作る
ここで、エディタを使用して以下のデータについては、blank.lis から除外してください。
AspgfTo_RH090523object038_kiki.fits
AspgfTo_RH090523object038_sheeta.fits
AspgfTo_RH090523object038_sophie.fits
AspgfTo_RH090523object039_ponyo.fits
AspgfTo_RH090523object039_san.fits
AspgfTo_RH090523object039_satsuki.fits
AspgfTo_RH090523object039_sheeta.fits
AspgfTo_RH090523object039_sophie.fits
AspgfTo_RH090523object042_sophie.fits
これらは、lAspgfTo_*.fits のデータがあります。
% blank.csh blank.lis

blank.csh を実行

ちなみに、
% cat blank.lis
AspgfTo_RH090523object038_chihiro.fits
AspgfTo_RH090523object038_clarisse.fits
AspgfTo_RH090523object038_fio.fits
...

マスク位置を示したファイル（練習データのディレクトリに入っています）
% ls blankmap_*
blankmap_lAspgfTo_RH090523object038_sheeta
blankmap_lAspgfTo_RH090523object038_sophie

% cat blankmap_lAspgfTo_RH090523object038_sheeta
1965　2030　2376　2552

ここで、各行は１個の四角形のマスクを意味します。各行４つの列からなり、x1 x2 y1 y2 の順で書きます。x=x1-x2, y=y1-y2 の四角形の領域がマスクされます。上記の例だと、x=1965-2030, y=2376-2552 の四角形がマスクされます。blankmap_* のファイルを作る時には IRAF の imexam を使うと便利です。(チェックしたい画像を ds9 に表示した状態で、cl> imexam とします。ds9 の画像上で b を二回押して四角形の２頂点を決めると位置が x1 x2 y1 y2 の順に出力されます。これをそのままカットアンドペーストして blankmap_* に書き込むことができます。)

実行後）
以下のような名前のファイル（マスクした画像）ができます。

bAspgfTo_RH090523object038_chihiro.fits
bAspgfTo_RH090523object038_clarisse.fits
bAspgfTo_RH090523object038_fio.fits
...

実行後のチェックポイント）
―　blankmap_*　のファイルで指定した位置がマスクされているはずです。マス

ク前の画像と比較してみてください。対応する blankmap_*　のファイルが無い画像は、何も処理されません。

（１１）組み合わせ規則作り(matching)

同じターゲット天体（天域）を複数回撮影した場合、複数の画像を重ね合わせて画質（S/N）を上げることができます。複数の画像を重ねる為には、画像間の位置の違いを知らなくてはいけません。また、異なる画像では天体からの flux が積分時間や空の透過率の違いによって違ってきてしまいます。このような画像間の位置や flux の違いを測定しておく作業が matching です。matching の作業では、各画像に写っている星（または stellar objects）を検出し、それらを元に各画像間の位置関係と flux 比を求めます。

コマンド：

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

ここで

・[makemos.lis]：matching を行う画像のリスト
・[starsel nskysigma]：何倍の skysigma が有意か（matching に使う星を選ぶ基準）
・[starsel npix]：有意な skysigma が何 pix 連なったとき天体と見なすか（matching に使う星を選ぶ基準）
・[starsel peakmin]：stellar object の flux peak の許容最小値（matching に使う星を選ぶ基準）
・[starsel peakmax]：stellar object の flux peak の許容最大値（matching に使う星を選ぶ基準）
・[aperture phot radius in pix]：測光に使う円形 aperture の半径の大きさ
・[output mos-file name]：chip 間の相対位置、感度が書かれた出力ファイルの名前

例）

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

matching をするオブジェクト画像のリストを作る

```
% makemos.csh makemos.lis 5 30 1000 40000 10 all.mos
```

makemos.csh を実行

ちなみに、
% cat makemos.lis
bAspgfTo_RH090523object038_chihiro.fits
bAspgfTo_RH090523object038_clarisse.fits
bAspgfTo_RH090523object038_fio.fits
...

実行後）
以下のような名前のファイルができる

all.mos
中身は、
% cat all.mos
bAspgfTo_RH090523object038_chihiro.fits 0.000000 0.000000 0.000000 1.000000
bAspgfTo_RH090523object038_clarisse.fits 2075.014139 1.531898 -0.000102 1.017556
bAspgfTo_RH090523object038_fio.fits 4184.051440 1.028001 -0.000054 1.079106
...

のように５列からなるリストです。
このファイルの意味は次のようになっています。まず、
makemos.lis の一番最初に書かれていた画像、
bAspgfTo_RH090523object038_chihiro.fits（今回の場合）が
基準になります（基準画像）。all.mos の２行目の情報は、この基準画像に対して、
bAspgfTo_RH090523object038_clarisse.fits はＸ方向（右が正）に 2075.0pix、
Y 方向（上が正）に 1.5pix、回転角が-0.000102radian（反時計回りが正）、flux 比が、1.018 倍であるという意味です。

実行後のチェックポイント）
― 上記のように all.mos が５列からなるリストになっていれば大抵の場合は成功していることを意味します。成功しなかった場合は、all.mos 自体が作られなかったり、作られてもエラーメッセージが含まれていたりおかしなリストになっていたりします。
出来上がった all.mos の中身は良く見てください。X,Y の位置や flux 比がリーズナブルになっているかどうか確認してください。これらの項目におかしな値が入っ

ている場合は、matching が失敗しています。

— all.mos の中身の大まかな確認方法には以下のような例が考えられます。各自で工夫されてよい方法を確立されたらぜひ開発チームにもお知らせください。

1．all.mos 内の 2 列目と 3 列目の値をそれぞれ縦軸・横軸としてプロットしてみます。この図は各ショットの基準画像からの位置ずれを表しているので、観測時のディザーパターンと一致するかどうかを見れば、all.mos の X、Y の出力がリーズナブルかどうかを判断できます。大きな外れ値がある場合は、そのチップの matching に失敗している可能性があります。

2．各チップ間の距離はショットごとにそれほど変わらずほぼ一定のはずなので、all.mos 内の 2 列目と 3 列目を使ってチップ間の距離を計算すれば、matching の失敗を判定出来ます。 たとえば、chihiro チップと sheeta チップとの間の距離が、とあるショットだけで他のショットでの値と大きく異なっていれば、そのショットのデータのマッチングに失敗している可能性があります。

3．all.mos 内の 5 列目の値（flux 比）を各ショット間で比較してみます。安定した天候条件下で取られたショット間の flux 比は、積分時間に比例していると考えられます。

4．all.mos を用いてモザイキングされた画像を直接目で確認することは matching が成功したかどうかの重要なチェック事項です。次節も参照してください。

— makemos.csh を実行中に、各画像に対して、

…
selected stars = 119
…

のように、検出された stellar object の個数が表示されます。この数が極端に少ない（３０個以下）または多い場合は all.mos 作りが失敗することがあります。次の４つのパラメター値を変えて調節してください。[starsel nskysigma]、[starsel npix]、[starsel peakmin]、および[starsel peakmax]です。matching が失敗する原因は、この４つのパラメータの取り方が悪いために適切な stellar object を画像から選び出せないことにあることが多いです。

（注意）１つの天体（天域）に対して画像が１ショット分しかない場合や、複数ショットの画像の組み合わせが必要ない場合はこの処理、および（１２）の処理を行う必要はありません。

（参考）次節のモザイキング（組み合わせ）過程では、all.mos ファイル内の値を用いて以下の公式に従い各入力ファイルの位置変換が行われます。

$$x\_mos = \cos(theta)\, x - \sin(theta)\, y + x\_local$$
$$y\_mos = \sin(theta)\, y + \cos(theta)\, y + y\_local$$

ここで、右辺の x、y が入力ファイルの変換される各ピクセルの座標で、左辺の x_mos と y_mos は、変換後の x, y 座標です。右辺の x_local と y_local は mos ファイルの 2 列目、3 列目（x, y オフセット）、theta は 4 列目（(x,y)=(0,0) を回転軸として x 軸から左回りに測った回転角度（radian））に対応しています。

（１２）組み合わせ(mosaicing)

手順（１１）の matching で求めた組合せ規則を元に、バラバラの画像を 1 枚に組合せ、足し合わせます。

コマンド：
% imcio2a [引数] [mos file] [result image]

ここで
・[引数]：imcio2a で使う引数(-dist_clip -nline=20 -dtype=FITSFLOAT -ignor=-32768)
・[mos file]：matching で作った組合せ規則のファイル
・[result image]：最終画像の名前

例）
% imcio2a -dist_clip -nline=20 -dtype=FITSFLOAT -pixignr=-32768 all.mos all.fits
imcio2a を実行

ちなみに、
% cat all.mos
bAspgfTo_RH090523object038_chihiro.fits 0.000000 0.000000 0.000000 1.000000
bAspgfTo_RH090523object038_clarisse.fits 2075.014139 1.531898 -0.000102 1.017556
bAspgfTo_RH090523object038_fio.fits 4184.051440 1.028001 -0.000054 1.079106
...

実行後）
以下のような名前のファイル（最終画像）ができる

all.fits

これが最終画像になります。この画像を元に、天体のカタログを作ったりするなど、サイエンティフィックな解析を行うことができます。

実行後のチェックポイント）

— 出来上がった最終画像を ds9 などで注意深く見てください。（１１）の matching がうまくいかなかった場合などは、星像が歪んでいたりすることがあります。

（注意１）makemos.lis の一番最初に書かれた画像が基準画像になることは説明しましたが、この基準画像の FITS header に記述されている World Coordinate System (WCS) が TAN 以外の場合でも、最終画像の WCS は強制的に TAN になります。この場合、最終画像の WCS は不正確なものになると思われます。WCS の情報が必要な際は、きちんと位置較正を行うようにしてください。例えば、SMOKA (http://smoka.nao.ac.jp) から配布されているアーカイブデータの中には、WCS が TNX になっているものが含まれていますので、特にご注意ください。

（注意２）makemos.csh では、入力されているデータの WCS は、(1)TAN で記述されていること、(2)大きく間違っていない位置情報が記述されていること、が前提になっています。WCS が TAN 以外で記述されている場合、makemos.csh が正しく動作しない場合がありますので、ご注意ください。

（参考１） [引数]の一つ -dist_clip は clipped mean による画像合成を意味します。-dist_clip の代わりに、-dist_med とすれば median による画像合成が、-dist_add とすれば mean による画像合成を作ることができます。また、-nline はモザイク画像作成において y 軸方向に一度に何ピクセル分処理するかを指定しています。ショット数が大きい場合、メモリ容量の比較的小さいコンピュータ環境では全画像をうまく扱えずコマンドが失敗することがありますが、-nline を小さく設定すれば、画像を細かく分割して処理することになりますので、コマンドが無事完了する可能性が高くなります。例では y 軸方向に 20 ピクセルずつ分割して画像を処理するよう指示しています。

（参考２）[引数]として、-dist_clip, -dist_med, -dist_add の代わりに –dist_peak を与えることで、最終画像のチェックを行うことが出来ます。このオプションは移動天体を検出

するためのもので、各ピクセルの(max)-(median)が出力されます。通常は出力画像の全視野内に数個の天体が見つかりますが、もしも特定の領域に多数の天体（明るいピクセル群）が見つかった場合には、その近辺でマッチングが失敗している可能性があります。

・ 標準星天体の処理

標準星データの処理は以下のようになります。以下の４段階の処理は全てターゲット画像の処理と同様です。重要な点は「フラットを作成する代わりに、ターゲット画像で用いたフラットをコピーしてくる」という点です

（S１）画像ファイル名の変換および画像の確認

コマンド：

% namechange.csh [SUP file names]

ここで

・[SUP file names]： 名前を変えたいファイル名のリスト

再度注意を喚起しますが、コマンドは、処理に使うファイルの置かれているディレクトリで実行しなければなりません。

例）

% cd standard

標準星データのあるディレクトリに移動

% ls -1 SUPA*.fits > namechange.lis

名前を変えるファイル名のリストを作る。

% namechange.csh namechange.lis

namechange.csh を実行

ちなみに、

% cat namechange.lis

SUPA01099710.fits

SUPA01099711.fits

SUPA01099712.fits

...

実行後）

SUPA...という名前のファイルは以下のようなファイル名に変わる

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

実行後のチェックポイント）
― SUPA...という名前のファイルがちゃんと H…という名前に変わっているはずです。ls コマンドで確認できます。

（S２）bias 引きおよび overscan の切り取り

standard ディレクトリ中で以下の操作を行ないます。

例）
% ls -1 H090*.fits > overscansub.lis
解析に使う画像のリストを作る。
% overscansub.csh overscansub.lis
overscansub.csh を実行

ちなみに、
% cat overscansub.lis
H090523object021_chihiro.fits
H090523object021_clarisse.fits
H090523object021_fio.fits
...

実行後）
以下のようなファイル(overscan bias 引きと切り取り済み画像）ができる。

To_RH090523object021_chihiro.fits
To_RH090523object021_clarisse.fits
To_RH090523object021_fio.fits
...

（S３）感度補正（flat fielding）

（３）で作成した flat 画像を用いて、１画像内での相対感度の補正をします。 なお、標準星画像と目標天体画像で使う flat 画像は、必ず同じものである必要があります。そこで、flat/dome_mflat* を標準星画像を置いたディレクトリにリンクしてから、コマンドを実行します。

例）

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

(3)で作ったフラットを standard ディレクトリにリンクします。

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

(3)で作ったフラットの名前が載ったリストを作る

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

感度補正を行う画像のリストを作る

% ffield.csh ffield_mf.lis ffield_im.lis

ffield.csh を実行

ちなみに、

% cat ffield_mf.lis

dome_mflat_chihiro.fits

dome_mflat_clarisse.fits

dome_mflat_fio.fits

...

% cat ffield_im.lis

To_RH090523object021_chihiro.fits

To_RH090523object021_clarisse.fits

To_RH090523object021_fio.fits

...

実行後）

以下のような名前のファイル（感度補正済みの画像）ができます

fTo_RH090523object021_chihiro.fits

fTo_RH090523object021_clarisse.fits

fTo_RH090523object021_fio.fits

...

（S４）歪補正(distortion correction)および微分大気差補正

例）

% ls -1 fTo_RH090*.fits > distcorr.lis

歪補正を行うオブジェクト画像のリストを作る

% distcorr.csh distcorr.lis

distcorr.csh を実行

ちなみに、

% cat distcorr.lis

fTo_RH090523object021_chihiro.fits

fTo_RH090523object021_clarisse.fits

fTo_RH090523object021_fio.fits

...

実行後）

以下のような名前のファイル（歪み補正済みの画像）ができます。

gfTo_RH090523object021_chihiro.fits

gfTo_RH090523object021_clarisse.fits

gfTo_RH090523object021_fio.fits

...

（S５）チップ間相対感度補正

（Ｓ４）が終った段階では標準星画像はチップ間での感度補正がされていません。例えば、練習画像を基に(１１)で作成した all.mos を見ると、以下のようになっています。

% cat all.mos

bAspgfTo_RH090523object038_chihiro.fits 0.000000 0.000000 0.000000 1.000000

bAspgfTo_RH090523object038_clarisse.fits 2075.014139 1.531898 -0.000102 1.017556

…

bAspgfTo_RH090523object038_ponyo.fits -0.083782 -4201.238704 -0.000492 0.805606

...

この all.mos から ponyo の感度は chihiro に比べて 80%程度であることが分かります。例

えば chihiro で 10000 ADU のカウントを持つ星は、pnoyo では 8000ADU 程度のカウントにしかなりません。このようなチップ間の相対感度比を、（１ 1）で作成した all.mos を基に補正する必要があります。
この処理は、標準星が chihiro にだけ写っているような場合は必要ありませんが、練習用データのようにいくつものチップに散っている場合に必要となります。
具体的には all.mos 中の該当するチップの感度で、標準星画像を割る操作が必要となります。

・ その他の重要な情報

2008 年 7 月に Suprime-Cam は CCD の交換を行いました。SDFRED 2 はこの新 CCD (FDCCD) で取得されたデータに対応したソフトです。2008 年 6 月より前に取得されたデータの解析には、SDFRED 1 をご利用ください。また、2008 年 7 月以降のデータについても、以下のような問題が知られています。

1) 2008 年 7 月 29 日から 2008 年 12 月 3 日の間に取得されたデータ
FRAMEID：SUPA01000001 - SUPA01055389
低いカウント値で、CCD のリニアリティに問題があることが知られています。リニアリティの誤差は、 <500ADU で 2-5% 以上になります。この期間のデータを扱う場合、カウント値が低いデータの使用は避けてください。この問題を解決するため、
2008 年 12 月に CCD の読み出しシステムの電圧設定を変更しました。2008 年 12 月 24 日以降（SUPA01155570-）のデータについては、リニアリティの問題はありません。

2) 2010 年 9 月 17 日に取得されたデータ
FRAMEID：SUPA01238890 – SUPA01240459
1 枚の CCD (DET-ID=9(san))について、一番左側のチャンネルでデータが正常に読み出せていません。この CCD の他のチャンネルのデータ、および他の CCD のデータは、正常です。この問題を解決するため、2010 年 10 月に san の一番左側のチャンネルについてのみ、読み出しシステムの電圧設定を変更しました。2010 年 10 月 5 日以降（SUPA01240460-）のデータについては、問題ありません。

3) 上記 1)、2)に記載されている通り、何度か読み出しシステムの電圧設定を変更しています。flat 作りを行う際、異なる電圧設定で取得されたデータを混ぜないように、気をつけてください。具体的には、以下の通りです。

全 CCD について
SUPA01000001 - SUPA01055389

SUPA01155570-

のデータは混ぜないこと。

DET-ID=9 (san) の CCD について

SUPA01155570 - SUPA01238889

SUPA01238890 - SUPA01240459

SUPA01240460 -

のデータは混ぜないこと。ただし、一番左側のチャンネル以外の３つのチャンネルについては SUPA0115570 – で flat のパターンは変化していないものと思われます。

4) 以下のデータは、FITS ヘッダーに誤りがあることが分かっています。

a) FRAMEID : SUPA01141740 – SUPA01141759 (2 ショット)
SUPA01196480 – SUPA01196489 (1 ショット)
これらのデータについては、位置情報 (RA, DEC, RA2000, DEC2000, CRVAL1, CRVAL2, CRPIX1, CRPIX2) が間違っています。他の FITS ヘッダー情報についても誤りがあると思われます。これらのデータは、SDFRED２ では正常に解析できないと思われますので、ご注意ください。

b) FRAMEID : SUPA01239571-SUPA01239630 (6 ショット)
これらのデータは、それぞれ SUPE01239571 のような下一桁が 1 の番号が EXP-ID になっており、SUPA01239571 - SUPA01239580 が同じ露出のセットとなっているなど、カウンターが一つずつずれている状態になっています。正しくは、EXP-ID は SUPE01239570 など下一桁が 0 の番号であるべきで、SUPA01239570 - SUPA01239579 が同じ露出のセットとなるべきです。なお、SUPA01239580 は chip 0 の画像となっているなど、FRAMEID の下一桁と DET-ID は一致しています。

SMOKA の以下のページにもデータに関する有用な情報が記載されています。
http://smoka.nao.ac.jp/about/subaru.jsp

謝辞：
SDFRED２に対する貴重なコメントやバグレポートを頂いた以下の方々に感謝の意を表します。
Masafumi Yagi (NAOJ), Gregory Zeimann (UC Davis), Ichi Tanaka (Subaru), Elinor Medezinski (Tel-Aviv Univ), Ryunosuke Imaeda (Tokyo Tech), Ken Mawatari (Tohoku Univ.)