pyIRSF user's manual

2014-03-11　Yasushi Nakajima

1. はじめに

IRSF+SIRIUS/SIRPOL で取得されたデータの 1 次処理をするために用意されているのが pyIRSF パッケージです。各種測定 (photometry や astrometry) ができるようなファイルを作成します。さらに、アパーチャー測光、2MASS カタログと比較して簡単な測光値の較正および (x, y) -> (RA, Dec) への変換、FITS への wcs の埋め込みをするアプリもあります。

1.1 必要な環境

OS　linux あるいは MacOSX のような UNIX 系 OS が必要です。なぜかというと pyIRSF パッケージは IRAF および PyRAF を利用するからです。

IRAF と PyRAF　PyRAF は IRAF のライブラリを Python から呼び出せるようにしたラッパーです。まず IRAF をインストールしてから PyRAF をインストールする必要があります。現時点での最新版 IRAF2.16.1 および PyRAF2.1.5 をおすすめします。インストール方法は私のウェブサイトで紹介していますので、そちらを参照してください。
http://irsf-software.appspot.com/yas/nakajima/pyraf.html

CentOS などのディストリビューションはシステムが Python に依存しています。2014 年 2 月現在では CentOS6.5 には Python2.6 がプレインストールされており、システムが Python2.6 に依存しています。Python2.6 でも pyIRSF パッケージは動きます。もし Python2.7 をインストールする場合には既存のものとかちあわないようにしてください。Web 上にそのような情報は多くありますのでそれらを参照にしてください。

Python　pyIRSF パッケージのスクリプトのほとんどは Python で書かれています。Linux の多くのディストリビューションや MacOSX には Python がプレインストールされています。もし入っていない場合には Python2.7 をインストールしてください。Python3 系には対応していません。

gcc コンパイラ　pyIRSF パッケージのスクリプトののこりは C 言語で書かれています。gcc コンパイラよりも上等なコンパイラでも動くとは思いますが試していません。

CFITSIO　SIRIUS の生データは、CFITSIO の fpack で圧縮されています。このファイルを読み込むために CFITSIO のライブラリが必要です。なお、pyIRSF パッケージを使う前に圧縮されたファイルを展開する必要はありません。CFITSIO のインストール方法は下を見てください。
http://irsf-software.appspot.com/yas/nakajima/linux/cfitsio.html

matplotlib　(オプション) Python のグラフ描画ライブラリです。トワイライトフラットからフラットフレームを作成した後に、変なフレームがないかを目視で確認する行程で使います。

pyIRSF パッケージ　以上のものがそろったら pyIRSF パッケージをダウンロードしてインストールしてください。
http://pyirsf.sourceforge.jp
からダウンロードできます。

1.2 インストール方法

/usr/local/bin で展開しないほうが良いと思います。各自のホームディレクトリで十分です。共有して使いたい場合には、せめて /usr/local あるいは /opt などがよいでしょう。

上記サイトからダウンロードした tar+gz ファイル (例 : pyIRSF-1.2.1.tar.gz) を適当な場所で展開します。すると pyIRSF-X.X(例 : oyIRSF-1.2.1) というディレクトリができます。X.X はバージョンです。そのディレクトリの中で **make** します。 cfitsio がないと怒られることがあるかもしれません。そのときには同じディレクトリの Makefile の LDFLAG と CFLAGS を編集してください。デフォルトでは /usr/local/lib および /usr/local/include に cfitsio のライブラリがあるものとしています。よくわからない人は中島に相談してください。

sirwcs.py はこのディレクトリにある OPM を呼び出します。既に OPM をインストールしている場合にも、ここで make してください。わざわざ pyIRSF の中の OPM にパスを通したりしなければかちあうこともありません。

sirwcs.py を使いたい場合には pyIRSF-X.X の中の OPM というディレクトリ内で **make** をしてください。一次処理後のフレームから星を検出して 2MASS カタログの星とのマッチングをするときに松永氏の OPM ソフトを利用します。

上記の作業が終わったら、pyIRSF-X.X にパスを通してください。あるいは、実行時に絶対パスをつけて呼び出すのでも構いません。相対パス (../../IRSF-1.2.1/sirius.py run など) で呼び出すとエラーが出ます。

1.3 SIRIUS の生データファイルと標準ディレクトリ構造

SIRPOL の場合ももちろん同様です。

jxxx.fits, hxxx.fits, kxxx.fits という三つのファイルをまとめて表す場合に、
[j|h|k]xxx.fits
のようにこのドキュメントでは表記します。

SIRIUS の生データのファイル名は
[j|h|k]yymmdd_nnnn.fits
のようになっています。最初の一文字がどのバンドかを示し、次に年月日 (2014 年 3 月 6 日なら 140306 です。)、そしてアンダースコアを挟んでその日の通し番号 4 桁が記され、最後に FITS ファイルであることを示す .fits の拡張子がつきます。
通常、これら生データは CFITSIO の fpack で圧縮されています。
[j|h|k]yymmdd_nnnn.fits.fz あるいは **[j|h|k]yymmdd_nnnn.fits.ic**
というように .fz あるいは .ic という拡張子がついた状態になっています。

これら生データは、その日付の yymmdd の名前がついたディレクトリの中の **rawdata** というディレクトリの中に入っています。これをこのドキュメントの中では **SIRIUS 標準ディレクトリ構造**と呼ぶ事にします。pyIRSF のデータ処理パイプラインではファイルやディレクトリがこの SIRIUS 標準ディレクトリ構造に従っているものと仮定しています。

2. pyIRSF パッケージ

pyIRSF パッケージは次のソフトウエアから成ります。

sirius.py　　SIRIUS のある日の生データセットから一次処理済みデータを作成するパイプラインソフトウエア。

sirpol.py　　SIRPOL のある日の生データセットから一次処理済みデータを作成するパイプラインソフトウエア。

mklog.py　　SIRIUS/SIRPOL のある日の生データセットから obslog ファイルを作成。

mktwflatlist.py　　トワイライトフラット観測フレーム群から、フラット補正ファイルを作成する元になるファイルセットを選び出し、そのリストを作成する。

twflatcom.py　　上記の mktwflatlist.py の作成したファイルを読み込み、フラット補正ファイルを作成する。C のソースコードから作成したバイナリ実行ファイル。

corrflat.py　　twflatcom.py で作成したフラット補正ファイルに、フラット光源の非一様成分についての補正を行う。

imgrecom.py　　sirius.py で処理したファイル群を、位置合わせして重ねる。sirius.py で重ね合わせに失敗したもののやり直しや、別の日のファイルを重ね合わせる際に使用する。

polrecom.py　　imgrecom.py の SIRPOL 版。

recombyhand.py　　視野内に星の数が少なくてどうしても上のソフトウエアでは重ならない場合に、これを使って目で見て各フレームの星を同定する。

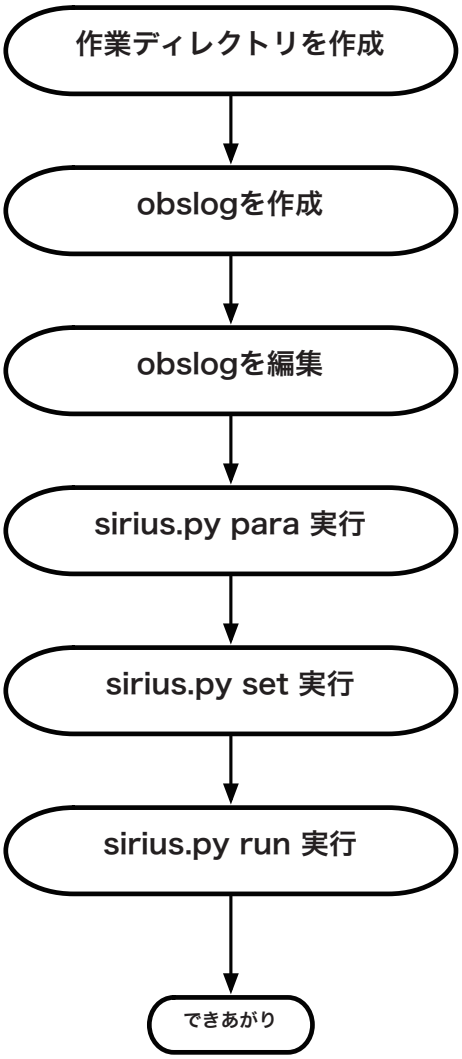
sirphot,py　　アパーチャー測光を行う

sirwcs.py　　アパーチャー測光を行い、2MASS カタログと比較して簡単な測光値較正を行い、x, y -> RA,DEC に変換してテキストファイルに結果を書き出す。さらに FITS ファイルに wcs を埋め込む。

3. SIRIUS のデータ処理 ~ 基本編

3.1 最も簡単な例

最も単純なケースではデータ処理の流れは以下のようになります。



実際の処理例を見ながら説明します。

2000 年 11 月 30 日のデータの中から、30Dor のデータだけを処理したい場合です。

```
$ pwd
/data/001130
$ ls
rawdata/
```

SIRIUS 標準ディレクトリ構造に従って、001130 のディレクトリの中に rawdata というディレクトリがあります。 まず行う事は、

(1) rawdata ディレクトリと同じ階層に作業ディレクトリを作成

です。作業ディレクトリの名前は何でもよいです。次に

(2) 作業ディレクトリの中で mklog.py を実行して obslog ファイルを作成

します。IRAF や PyRAF を起動してそのコマンドラインで入力するわけでは**ありません**。普通にシェルのコマンドラインで入力してください。以下同様です。

```
$ mklog.py
```

```
$ ls
rawdata/  wrk/
$ cd wrk
$ ls
obslog
```

この段階では、例えば、上のようになっています。

obslog はテキストファイルです。フレーム番号、オブジェクト名、積分時間、ディザリング時の赤経方向のオフセット、赤緯方向のオフセット、などがフレーム番号順に記載されています。mklog.py が FITS ヘッダの情報を読み込んでこれを作成します。

現状では、rawdata ディレクトリの中にあるすべてのファイルの情報が obslog の中にあります。このままでは全てのファイルの処理が行われます。30Dor の処理だけしたいので

(3) obslog ファイルの編集

を行います。

0116-0124 で 30Dor および 0125-0133 で 30Dorsky(30Dor のスカイバイアス補正フレーム) の観測がどちらも積分時間 30 秒で行われています。いわゆる 20 秒 x ディザリング 9 枚のターゲット 1 セット + スカイ 1 セットの観測です。

積分時間 30 秒のフレームの処理には、30 秒の dark のフレームも必要です。0065-0074 で取得されています。従って、obslog ファイルには 0065-0074, 0116-0133 の行だけを残して残りの行は emacs や vi などのテキストエディタを用いて削除します。

すると、obslog の中身は下のようになります。

一番最初の FRAME OBJECT .... が並んでいる行は削除しても構いません。

```
FRAME OBJECT ITIME RA_OFF DEC_OFF DATE_UTC TIME_UTC DATE_LT TIME_LT JD EPOCH RA DEC AIRMASS
0065 DARK 30. 0.0 0.0 2000/11/30 18:34:20.0 2000/11/30 20:34:20.0 51878.0 2000 00:33:15.2 -39:24:10.0 1.007650
0066 DARK 30. 0.0 0.0 2000/11/30 18:34:20.0 2000/11/30 20:34:20.0 51878.0 2000 00:33:15.2 -39:24:10.0 1.007650
0067 DARK 30. 0.0 0.0 2000/11/30 18:34:20.0 2000/11/30 20:34:20.0 51878.0 2000 00:33:15.2 -39:24:10.0 1.007650
0068 DARK 30. 0.0 0.0 2000/11/30 18:34:20.0 2000/11/30 20:34:20.0 51878.0 2000 00:33:15.2 -39:24:10.0 1.007650
0069 DARK 30. 0.0 0.0 2000/11/30 18:34:20.0 2000/11/30 20:34:20.0 51878.0 2000 00:33:15.2 -39:24:10.0 1.007650
0070 DARK 30. 0.0 0.0 2000/11/30 18:34:20.0 2000/11/30 20:34:20.0 51878.0 2000 00:33:15.2 -39:24:10.0 1.007650
0071 DARK 30. 0.0 0.0 2000/11/30 18:34:20.0 2000/11/30 20:34:20.0 51878.0 2000 00:33:15.2 -39:24:10.0 1.007650
0072 DARK 30. 0.0 0.0 2000/11/30 18:34:20.0 2000/11/30 20:34:20.0 51878.0 2000 00:33:15.2 -39:24:10.0 1.007650
0073 DARK 30. 0.0 0.0 2000/11/30 18:34:20.0 2000/11/30 20:34:20.0 51878.0 2000 00:33:15.2 -39:24:10.0 1.007650
0074 DARK 30. 0.0 0.0 2000/11/30 18:34:20.0 2000/11/30 20:34:20.0 51878.0 2000 00:33:15.2 -39:24:10.0 1.007650
0116 30Dor 30. 0.0 0.0 2000/11/30 20:04:19.4 2000/11/30 22:04:19.4 51878.0 2000 05:38:42.4 -69:06:03.0 1.464853
0117 30Dor 30. 14.6 3.9 2000/11/30 20:05:13.4 2000/11/30 22:05:13.4 51878.0 2000 05:38:45.2 -69:05:59.1 1.462924
0118 30Dor 30. 7.5 13.1 2000/11/30 20:06:06.2 2000/11/30 22:06:06.2 51878.0 2000 05:38:43.8 -69:05:49.8 1.460879
0119 30Dor 30. -3.9 14.6 2000/11/30 20:06:59.0 2000/11/30 22:06:59.0 51878.0 2000 05:38:41.7 -69:05:48.4 1.458846
0120 30Dor 30. -13.1 7.5 2000/11/30 20:07:52.9 2000/11/30 22:07:52.9 51878.0 2000 05:38:40.0 -69:05:55.3 1.456820
0121 30Dor 30. -14.6 3.9 2000/11/30 20:08:45.4 2000/11/30 22:08:45.4 51878.0 2000 05:38:39.7 -69:05:58.9 1.454902
0122 30Dor 30. -7.5 -13.1 2000/11/30 20:09:37.5 2000/11/30 22:09:37.5 51878.0 2000 05:38:41.0 -69:06:15.9 1.453108
0123 30Dor 30. 3.9 -14.6 2000/11/30 20:10:30.2 2000/11/30 22:10:30.2 51878.0 2000 05:38:43.2 -69:06:17.4 1.451285
0124 30Dor 30. 13.1 -7.5 2000/11/30 20:11:23.8 2000/11/30 22:11:23.8 51878.0 2000 05:38:44.9 -69:06:10.5 1.449398
0125 30Dorsky 30. 0.0 0.0 2000/11/30 20:16:43.6 2000/11/30 22:16:43.6 51878.0 2000 05:41:52.1 -69:05:55.0 1.444610
0126 30Dorsky 30. 19.4 5.2 2000/11/30 20:17:35.9 2000/11/30 22:17:35.9 51878.0 2000 05:55:26.1 -69:05:57.3 1.472587
0127 30Dorsky 30. 10.0 17.4 2000/11/30 20:18:28.8 2000/11/30 22:18:28.8 51878.0 2000 05:55:24.3 -69:05:45.2 1.47046
0128 30Dorsky 30. -5.2 19.4 2000/11/30 20:19:21.6 2000/11/30 22:19:21.6 51878.0 2000 05:55:21.5 -69:05:43.6 1.46834
0129 30Dorsky 30. -17.4 10.0 2000/11/30 20:20:14.1 2000/11/30 22:20:14.1 51878.0 2000 05:55:19.2 -69:05:52.8 1.4662
0130 30Dorsky 30. -19.4 5.2 2000/11/30 20:21:07.2 2000/11/30 22:21:07.2 51878.0 2000 05:55:18.8 -69:05:57.6 1.46430
0131 30Dorsky 30. -10.0 -17.4 2000/11/30 20:21:59.7 2000/11/30 22:21:59.7 51878.0 2000 05:55:20.6 -69:06:20.3 1.462
0132 30Dorsky 30. 5.2 -19.4 2000/11/30 20:22:52.5 2000/11/30 22:22:52.5 51878.0 2000 05:55:23.4 -69:06:22.2 1.46062
0133 30Dorsky 30. 17.4 -10.0 2000/11/30 20:23:46.0 2000/11/30 22:23:46.0 51878.0 2000 05:55:25.7 -69:06:12.8 1.4587
```

ここで、ターゲットのセットに少なくとも一つのスカイのセットが (観測順として) 隣接していることが重要です。オブジェクト名の最後に sky(大文字小文字関係なし) がついていればそのフレームはスカイの観測であると、パイプラインの中では認識されます。

(4) sirius.py para を実行

作業ディレクトリ内で実行します。

```
$ sirius.py para
yymmdd is 001130
the mastar flat frames are deployed
```

そのデータの日付を確認するメッセージがでます。2 番目のメッセージは何でしょう？今回はフラット補正フレームを用意しませんでした。sirius.py para コマンドはそのことをお見通しです。作業ディレクトリにフラット補正フレームと思しき FITS ファイルがないので、pyIRSF パッケージが用意しているマスターフラットをフラット補正フレームとして使う事を宣言したのです。[j|h|k]mflat.fits@ がそれらマスターフラット (へのシンボリックリンク) です。

マスターフラットで良いのでしょうか？この時期のものなら良いと考えています。後ほどその説明をします。

```
$ ls
hmflat.fits@    jmflat.fits@    kmflat.fits@  obslog obslog~  sirius.param
```

さらに、sirius.param というパラメータ設定ファイルができています。その中のパラメータをテキストエディタで編集することで、この後のパイプライン処理への細かい指令を出す事ができます。今回は単純なケースなのでその必要はありません。

(5) sirius.py set を実行

```
$ sirius.py set
```

この後のパイプライン処理に必要なファイルが作成されます。

(6) sirius.py run を実行

```
$ sirius.py run
```

パイプライン処理の開始です。処理のどの段階まで終わったか、各フレームのバックグラウンドレベルや星の fwhm、ディザリングしたフレームの位置合わせの変換の結果などがメッセージとして表示されます。見ているだけでいいです。

```
dark frames ...  have been created
dark subtraction and flat division ...  done
making sky bias frames and bad pixel masks ...  done.
masking ...  done
sky subtraction ...  j h k done

******************
o0.30Dor
******************
j-band
find stars for geomap/geotran
fnum fwhm ellip star_num thresh    med    stddev
0116  3.5  0.12   30      10.0     715.5  24.4
0117  3.7  0.12   30      10.0     704.4  24.5
0118  3.6  0.11   30      10.0     700.9  24.5
0119  3.8  0.10   30      10.0     699.2  24.5
0120  3.8  0.12   30      10.0     694.7  24.6
0121  3.7  0.10   30      10.0     691.1  24.6
0122  3.3  0.09   30      10.0     697.9  24.5
0123  3.7  0.15   30      10.0     688.4  24.4
0124  3.5  0.11   30      10.0     687.0  24.4
star match and geomap

frame     rot      x (rms)       y (rms)  num  (rnum/inum)  dx    dy
jf0116 : reference frame
jf0117   0.01   31.3 (0.18)  -10.0 (0.10) 198  (247/287)     -1.1  -1.3
jf0118   0.00   14.6 (0.15)  -30.2 (0.15) 198  (246/326)     -2.1  -1.1
jf0119  -0.00  -10.6 (0.15)  -32.3 (0.11) 201  (244/293)     -1.9   0.1
jf0120  -0.01  -30.0 (0.11)  -16.4 (0.12) 191  (246/274)     -0.9   0.3
jf0121  -0.01  -32.9 (0.14)   -8.5 (0.14) 198  (247/298)     -0.5   0.2
jf0122  -0.01  -15.5 (0.16)   28.7 (0.15) 195  (245/309)      1.2  -0.4
jf0123   0.00    9.2 (0.13)   30.0 (0.08) 199  (246/296)      0.5  -2.4
jf0124   0.01   28.8 (0.11)   14.4 (0.13) 197  (247/249)     -0.3  -2.3
geotran in progress
100% done
combining
done
```

... 途中省略 ...

```
kf0119   0.00  -10.2 (0.17)  -32.2 (0.13) 103  (131/117)     -1.5   0.2
kf0120  -0.01  -29.6 (0.18)  -16.6 (0.14) 109  (133/120)     -0.5   0.1
kf0121  -0.01  -32.6 (0.17)   -8.7 (0.14) 114  (134/129)     -0.2  -0.0
kf0122  -0.01  -15.6 (0.18)   28.6 (0.16) 118  (133/203)      1.1  -0.5
kf0123   0.00    9.1 (0.11)   30.2 (0.10) 100  (132/111)      0.4  -2.2
kf0124   0.01   28.6 (0.15)   14.7 (0.13) 110  (129/133)     -0.5  -2.0
geotran in progress
100% done
combining
done

cleaning the working directory...
done
```

(7) 結果を見てみる

処理が終わると作業ディレクトリの中は下のようになっています。

```
dithlistlist  hmflat.fits@  jmflat.fits@  kmflat.fits@  o0.30Dor/      obslog   sirius.param
ffiles/       hskybg.list   jskybg.list   kskybg.list   objectskylist  obslog~
```

一般には o0.[オブジェクト名] というディレクトリができます。o0. なんていうヘンテコなものがアタマについていますが、この後、複数のターゲットを同時に処理するときにその謎 (?) がとけます。

一番欲しいものは **o0.30Dor** のディレクトリの中にあります。

```
hquality.txt  htransummary.txt  jrecom.fits        kquality.txt  ktransummary.txt
hrecom.fits   jquality.txt      jtransummary.txt   krecom.fits
```

ここの [j|h|k]recom.fits が一番欲しいものです。各生データからダークを引き、フラット補正し、スカイバイアスを引き、全てのフレームの位置合わせをして重ね合わせたもの (average) です。

では、ds9 で見てみましょう。

```
$ ds9 jrecom.fits
```

ds9 の File > Display Header で FITS ヘッダを見てみましょう。
NAXIS1 と NAXIS2 は 1024 よりも大きくなっています。ディザリングして観測したものを位置合わせして重ねたからです。OBJECT から HUMIDITY までの行は、そのセットの一枚目の観測の情報です。特に時刻およびエアマスについては一枚目の積分開始時のものであるということに注意してください。
最後の 4 行にパイプライン処理時の情報が入っています。
NCOMBINE ： 何枚重ね合わせたか
COMMETHO : IMCOMBINE 時の重ね合わせの方法 (ここでは average のみ)
COMAREA : 全てのフレームが重なっている領域。ただし 1024 x 1024 全てのピクセルが有効であった場合。
REJECT : IMCOMBINE 時の reject 方法

実はこの FITS ファイルにはもう一つの画像が含まれています。

```
$ ds9 jrecom.fits[1]
```

これは各ピクセルが何枚の平均から計算されているかを示します。バッドピクセル等がなければ上記の COMAREA の情報で十分なのですが、フレームの端っこのほうには、特に J バンドで、バッドピクセルがかたまって存在するため、COMAREA だけでは十分ではありません。一般に、同じフレーム上でも、この数が異なる領域では等級のゼロ点が異なる事に注意してください。

このディレクトリには FITS ファイルだけではなく、テキストファイルもあります。

同じセットなのに、fwhm や ellip が極端に違うフレーム、あるいは星の数が極端に少ないフレームが見つかる場合があります。そのような時はそのフレームを目で確かめて、本当に変だったら (星がのびている、おそらく雲が通って星が少ない)、そのフレームを含めずに重ね合わせをしなおすのがよいでしょう。そういう場合には、重ね合わせの行程だけをやりなおす imgrecom.py を使います。

[j|h|k]quality.txt ： この内容はパイプライン処理時にも標準出力に表示されたものです。
重ね合わせる前の個々のフレームでの、星の fwhm(pixel) と ellipticity、バックグラウンドのメジアン、標準偏差が記録されています。star_num および thresh というカラムもあります。star_num は fwhm と ellip を計算するために使った星の数です。検出された星の中で最も明るい 30 個を通常使います。したがって、ここには 30 以上の数値は入りません。thresh は星の検出のときの iraf.daofind のシグマの値です。デフォルトは 10 です。あまり星の数が多い (>250) 場合にはこの数を自動的に上げます。

[j|h|k]transummary.txt ：これもパイプライン処理時にも標準出力に表示されたものです。

```
frame     rot      x (rms)       y (rms)  num  (rnum/inum)  dx    dy
hf0116 : reference frame
hf0117   0.01    31.3 (0.17)  -10.0 (0.09) 173  (217/192)     -1.1  -1.3
hf0118   0.00    14.7 (0.12)  -30.2 (0.12) 183  (216/215)     -2.0  -1.1
hf0119  -0.00   -10.5 (0.16)  -32.3 (0.11) 164  (220/183)     -1.8   0.1
hf0120  -0.01   -30.0 (0.11)  -16.4 (0.12) 166  (221/185)     -0.9   0.3
hf0121  -0.01   -32.9 (0.10)   -8.6 (0.12) 177  (221/197)     -0.5   0.1
hf0122  -0.01   -15.5 (0.15)   28.6 (0.14) 110  (220/125)      1.2  -0.5
hf0123   0.00     9.2 (0.13)   30.1 (0.10)  91  (218/104)      0.5  -2.3
hf0124   0.01    28.8 (0.10)   14.5 (0.12) 110  (216/127)     -0.3  -2.2
```

1 枚目のフレーム (referecnce frame) に対してそれぞれのフレーム (input frame) がどのくらい x, y 方向にシフトおよび回転しているかの情報が記載されています。

rms の値が大きい場合 (>0.5) はフレーム間の星のマッチングが失敗しているかもしれません。dx, dy については、望遠鏡のポインティングが徐々にずれていくことで徐々に大きくなる事があります。前後のフレームと比べて大きく違う (差が 5-10 以上など) ときにはマッチングが失敗している可能性が高いです。

rot : 回転角 (度)　　**x, y** ： reference の星の位置に対して input frame の星が何ピクセル x, y 方向にシフトして写っているかの平均値。　　**rms** : x, y 平均値からのばらつき
num : マッチできた星の数。
rnum/inum ： そのマッチを行うときに使われた rnum = reference frame での星の数。inum = input frame での星の数。
dx, dy ： 望遠鏡のシフト情報から期待される x,y からどの程度ずれていたか。

作業ディレクトリの中にはターゲットの重ね合わせの結果のディレクトリだけではなく、**ffiles** というディレクトリもあります。ここには、ダーク引き、フラット補正、およびスカイバイアス引きが済んだ、重ね合わせ前の個々のフレームが保存されています。[j|h|k]fnnnn.fits という名前がついています。
重ね合わせをやり直す場合、重ねる前の個々のフレームのクォリティをチェックする場合、などのために保存してあります。なお、スカイバイアスフレームのセットについても、セルフスカイを引いたものが保存されています。

作業ディレクトリの中には、以上の他に、新しく作成されたファイルとして **[j|h|k]skybg.list** というファイルがあります。このテキストファイルには、スカイバイアスフレームのバックグラウンドのメジアン値が記録されています。

3.2 複数のターゲットを処理

2000 年 11 月 30 日のデータの中から、LMCcent, LMCcent2, LMCcent3 のデータを処理したい場合です。今度は、3 つの違う視野を観測したセットを一気に処理します。流れは上の例と同じです。

(1) rawdata ディレクトリと同じ階層に作業ディレクトリを作成
さきほどの作業ディレクトリの名前と重ならないように。
(2) 作業ディレクトリの中で mklog.py を実行して obslog ファイルを作成
(3) obslog ファイルの編集
0065-0073 の 30 秒のダークのセット、および 0134-0178 の LMCcent - LMCcentsky - LMCcent2 - LMCcent2sky - LMCcent3 のセットを残して残りは削除します。
target - sky-target - sky - target となっており、target のセットが少なくとも一つの sky のセットと隣接しているようにしてください。そうでないと sirius.py set のときに警告がでます。
(4) sirius.py para を実行
(5) sirius.py set を実行
(6) sirius.py run を実行
(7) 結果を見てみる

```
dithlistlist  hskybg.list   kmflat.fits@  o2.LMCcent2/   obslog
ffiles/       jmflat.fits@  kskybg.list   o4.LMCcent3/   obslog~
hmflat.fits@  jskybg.list   o0.LMCcent/   objectskylist  sirius.param
```

o[整数]. の整数がとびとびですね。これは途中に sky セットの観測があったためです。途中の処理の都合上、スカイセットにもセットとしての通し番号がふられます。

こんな感じです。ターゲット毎に o[整数].[オブジェクト名] のディレクトリができています。その中に FITS ファイルやテキストファイルが保存されています。o[整数]. はターゲットを区別するためのインデックスだと思ってください。オブジェクト名が同じでも別のセットとして分けて重ね合わせたい場合もあり得るので、このようなインデックスをつけることにしました。

3.3 ターゲットとスカイの組み合わせを変更する

上の例で (5) の sirius.py set を実行した後に作成される objectskylist を見てみましょう。

```
0134-0142 LMCcent 0143-0151 sky0
0143-0151 LMCcentsky 0143-0151 sky0
0152-0160 LMCcent2 0161-0169 sky2
0161-0169 LMCcent2sky 0161-0169 sky2
0170-0178 LMCcent3 0161-0169 sky2
```

このファイルの中には、フレームのセットとスカイセットの対応が記述されています。最初の行には、「0134-0142 のフレームセットは LMCcent というオブジェクト名なんだけど、これ

らのフレームからスカイバイアスを引くときには、0143-0152 のセットから作成されたスカイバイアスを使います。ちなみに、仮に sky0 と名付けておきます。」と書かれてあります。
最後の行は、0170-0178 の LMCcent3 についての記述があります。LMCcent3sky という観測セットはありませんが、直前に観測された LMCcent2sky のセットから作られるスカイバイアスとの対応付けがなされています。

さて、3 行目の 0152-0160 の LMCcent2 のセットについての記述についてです。これらのフレームからは 0161-0169 の LMCcent2sky のセットから作成されるスカイバイアスを引くことになっています。ですが、直前の 0143-0151 の LMCcentsky のセットも使うほうがより良いスカイバイアスがつくれるのではないでしょうか？前後のセットから平均的なスカイバイアスを作る事で、LMCcent2 の観測時のスカイバイアスがより良く再現できそうです。
そのためには、(5) sirius.py set の実行の後に、
(5') objectskylist の編集
(5'') sirius.py revsky の実行
を行ってください。
ここでの例の場合の objecyskylist の編集方法は以下の通りです。

```
0152-0160 LMCcent2 0161-0169 sky2
```

の部分を

```
0152-0160 LMCcent2 0143-0151:0161-0169 sky3
```

と書き換えます。0161-0169 に加えて 0143-0151 のセットも取り入れます。また、その結果、新たに作成されるスカイバイアスの名前を新しいものに変えます。
その編集の後に、sirius.py revsky を実行すると同じ作業ディレクトリ内にある関連テキストファイルが更新されます。それから
(6) sirius.py run を実行
(7) 結果を見てみる
という行程を踏みます。

ここで sky は大文字でも小文字でも構いません。

[objectskylist 自動生成のルール]
AAA という object 名のターゲットの観測セットがあるとします。
1. 観測順として前後に隣接する観測セットの object 名が AAAsky である場合。
   その (前後両方であればそれら両方の) 観測セットから AAA のセット用のスカイバイアスフレームが作られます。
2. 観測順として前後に隣接する観測セットの object 名がどちらも AAAsky でない場合。
   前後のどちらか一方のみのセットの object 名が sky で終わる場合、そのセットから AAA のセット用のスカイバイアスフレームが作られます。前後の両方のセットの object 名が sky で終わる場合、前のセットから AAA のセット用のスカイバイアスフレームが作られます。
3. 観測順として前後に隣接する観測セットの object 名がどちらも sky で終わらない場合。
   sirius.py set 実行後に　No sky for AAA というメッセージが出ます。何も処理は行われません。

3.4 同じターゲットを複数のセットにわたって観測した

総積分時間を増やすためにこのような観測は頻繁に行われます。その場合、パイプライン処理でそれら全てのセットの観測を一つのファイルに重ね合わせる必要があります。

2000 年 11 月 30 日のデータの中から ngc2071 のデータを処理します。この夜には同じ ngc2071 の視野を 4 つのセットで観測しています。

(1) rawdata ディレクトリと同じ階層に作業ディレクトリを作成

(2) 作業ディレクトリの中で mklog.py を実行して obslog ファイルを作成

ここまでは同じパターンです。

(3) obslog ファイルの編集

0065-0073 の 30 秒のダークのセット、0189-0251 の ngc2071n1 - ngc2071n1sky - ngc2071n2 - ngc2071n2sky - ngc2071n3 - ngc2071n3sky - ngc2071n4 のターゲットとスカイの観測を繰り返す 7 つのセットのみを残して残りは削除します。

ここで！ ngc2071n1 というように n1 などの記号は別の観測セットですよという意味でつけられているのですが、この ngc2071n? および ngc2071n?sky についている n1, n2, n3, n4 を obslog の中から消します。

長いので全部見せませんが、編集の結果 obslog は

```
FRAME OBJECT ITIME RA_OFF DEC_OFF DATE_UTC TIME_UTC DATE_LT TIME_LT JD EPOCH RA DEC AIRMASS
0065 DARK 30. 0.0 0.0 2000/11/30 18:34:20.0 2000/11/30 20:34:20.0 51878.0 2000 00:33:15.2 -39:24:10.0 1.007650
0066 DARK 30. 0.0 0.0 2000/11/30 18:34:20.0 2000/11/30 20:34:20.0 51878.0 2000 00:33:15.2 -39:24:10.0 1.007650
0067 DARK 30. 0.0 0.0 2000/11/30 18:34:20.0 2000/11/30 20:34:20.0 51878.0 2000 00:33:15.2 -39:24:10.0 1.007650
0068 DARK 30. 0.0 0.0 2000/11/30 18:34:20.0 2000/11/30 20:34:20.0 51878.0 2000 00:33:15.2 -39:24:10.0 1.007650
0069 DARK 30. 0.0 0.0 2000/11/30 18:34:20.0 2000/11/30 20:34:20.0 51878.0 2000 00:33:15.2 -39:24:10.0 1.007650
0070 DARK 30. 0.0 0.0 2000/11/30 18:34:20.0 2000/11/30 20:34:20.0 51878.0 2000 00:33:15.2 -39:24:10.0 1.007650
0071 DARK 30. 0.0 0.0 2000/11/30 18:34:20.0 2000/11/30 20:34:20.0 51878.0 2000 00:33:15.2 -39:24:10.0 1.007650
0072 DARK 30. 0.0 0.0 2000/11/30 18:34:20.0 2000/11/30 20:34:20.0 51878.0 2000 00:33:15.2 -39:24:10.0 1.007650
0073 DARK 30. 0.0 0.0 2000/11/30 18:34:20.0 2000/11/30 20:34:20.0 51878.0 2000 00:33:15.2 -39:24:10.0 1.007650
0074 DARK 30. 0.0 0.0 2000/11/30 18:34:20.0 2000/11/30 20:34:20.0 51878.0 2000 00:33:15.2 -39:24:10.0 1.007650
0189 ngc2071 30. 0.0 0.0 2000/11/30 22:14:32.0 2000/12/01 00:14:32.0 51879.0 2000 05:47:04.5 +00:21:44.3 1.2847
0190 ngc2071 30. 19.4 5.2 2000/11/30 22:15:29.4 2000/12/01 00:15:29.4 51879.0 2000 05:47:05.8 +00:21:49.3 1.282
0191 ngc2071 30. 10.0 17.4 2000/11/30 22:16:27.4 2000/12/01 00:16:27.4 51879.0 2000 05:47:05.1 +00:22:01.5 1.28
0192 ngc2071 30. -5.2 19.4 2000/11/30 22:17:24.1 2000/12/01 00:17:24.1 51879.0 2000 05:47:04.1 +00:22:03.6 1.27
0193 ngc2071 30. -17.4 10.0 2000/11/30 22:18:21.3 2000/12/01 00:18:21.3 51879.0 2000 05:47:03.3 +00:21:54.6 1.27
0194 ngc2071 30. -19.4 5.2 2000/11/30 22:19:18.4 2000/12/01 00:19:18.4 51879.0 2000 05:47:03.2 +00:21:49.4 1.27
0195 ngc2071 30. -10.0 -17.4 2000/11/30 22:20:15.7 2000/12/01 00:20:15.7 51879.0 2000 05:47:03.8 +00:21:27.0 1.2
0196 ngc2071 30. 5.2 -19.4 2000/11/30 22:21:12.5 2000/12/01 00:21:12.5 51879.0 2000 05:47:04.8 +00:21:25.0 1.27
0197 ngc2071 30. 17.4 -10.0 2000/11/30 22:22:09.6 2000/12/01 00:22:09.6 51879.0 2000 05:47:05.6 +00:21:34.1 1.28
0198 ngc2071sky 30. 0.0 0.0 2000/11/30 22:25:24.4 2000/12/01 00:25:24.4 51879.0 2000 05:48:12.2 +00:21:43.0 1.28
0199 ngc2071sky 30. 19.4 5.2 2000/11/30 22:26:21.1 2000/12/01 00:26:21.1 51879.0 2000 06:03:45.7 +00:20:36.3 1.2
```

... 途中省略 ...

```
0238 ngc2071sky 30. -17.4 10.0 2000/11/30 23:15:06.1 2000/12/01 01:15:06.1 51879.0 2000 05:37:03.3 +00:22:37.7 1.2
0239 ngc2071sky 30. -19.4 5.2 2000/11/30 23:16:03.3 2000/12/01 01:16:03.3 51879.0 2000 05:37:03.2 +00:22:32.9 1.19
0240 ngc2071sky 30. -10.0 -17.4 2000/11/30 23:17:00.2 2000/12/01 01:17:00.2 51879.0 2000 05:37:03.8 +00:22:10.2 1.
0241 ngc2071sky 30. 5.2 -19.4 2000/11/30 23:17:57.7 2000/12/01 01:17:57.7 51879.0 2000 05:37:04.8 +00:22:08.0 1.19
0242 ngc2071sky 30. 17.4 -10.0 2000/11/30 23:18:55.5 2000/12/01 01:18:55.5 51879.0 2000 05:37:05.6 +00:22:17.8 1.2
0243 ngc2071 30. 0.0 0.0 2000/11/30 23:22:09.6 2000/12/01 01:22:09.6 51879.0 2000 05:47:04.5 +00:21:44.1 1.193901
0244 ngc2071 30. 19.4 5.2 2000/11/30 23:23:06.8 2000/12/01 01:23:06.8 51879.0 2000 05:47:05.8 +00:21:49.2 1.19347
0245 ngc2071 30. 10.0 17.4 2000/11/30 23:24:04.5 2000/12/01 01:24:04.5 51879.0 2000 05:47:05.1 +00:22:01.7 1.1930
0246 ngc2071 30. -5.2 19.4 2000/11/30 23:25:01.0 2000/12/01 01:25:01.0 51879.0 2000 05:47:04.1 +00:22:03.5 1.1926
0247 ngc2071 30. -17.4 10.0 2000/11/30 23:25:58.2 2000/12/01 01:25:58.2 51879.0 2000 05:47:03.3 +00:21:54.2 1.192
0248 ngc2071 30. -19.4 5.2 2000/11/30 23:26:55.3 2000/12/01 01:26:55.3 51879.0 2000 05:47:03.2 +00:21:49.3 1.1918
0249 ngc2071 30. -10.0 -17.4 2000/11/30 23:27:52.3 2000/12/01 01:27:52.3 51879.0 2000 05:47:03.8 +00:21:26.6 1.19
0250 ngc2071 30. 5.2 -19.4 2000/11/30 23:28:49.7 2000/12/01 01:28:49.7 51879.0 2000 05:47:04.8 +00:21:24.7 1.1911
0251 ngc2071 30. 17.4 -10.0 2000/11/30 23:29:48.4 2000/12/01 01:29:48.4 51879.0 2000 05:47:05.6 +00:21:34.2 1.1901
```

のようになります。

(4) sirius.py para を実行

(5) sirius.py set を実行

ここで objectskylist が作成されます。この例の場合、観測順として前後両方に隣接したスカイバイアス観測セットが使われます。(もちろん前や後になければ片方だけです)

(6) sirius.py run を実行

(7) 結果を見てみる

作業ディレクトリの中は以下のようになります。

```
dithlistlist  hskybg.list   kmflat.fits@  objectskylist  sirius.param
ffiles/       jmflat.fits@  kskybg.list   obslog
hmflat.fits@  jskybg.list   o0.ngc2071/   obslog~
```

ここには o0.ngc2071 のディレクトリしかありません。object 名の同じセットは同じ視野を観測していると認識され、全て一つに重ねたものができます。

3.5 フラット補正 FITS ファイルの準備

ここまでは、フラット補正 FITS ファイルについては、pyIRSF パッケージが準備しているマスターフラットを使う前提で話をすすめてきました。マスターフラットは、少なくとも 2000 年から 2004 年のデータ処理に使う分には全く問題がありません。これまで調査した分では、2008 年までのデータについても 1% 以内の精度で問題ないことがわかっています。それ以降についてはまだ調査していないだけなのですか、根拠もなしに使うのも気持ち悪いので、その時期のデータに適したフラット補正 FITS ファイルを作成して使用しましょう。

フラット補正 FITS ファイルの作成については、**5. フラット補正 FITS ファイルの作成**を参考にしてください。pyIRSF パッケージの中の mktwflist.py、twfcom.py および corrflat.py を使用します。

独自のフラット補正 FITS ファイルができたら、その FITS ファイルを作業ディレクトリの中にコピーしてください。順番として、obslog の作成および編集の前でも後でも構いません。ただし、**`sirius.py para` の実行の前**には作業ディレクトリにフラット補正 FITS ファイルがあるようにしてください。 sirius.py para の処理が、作業ディレクトリにある FITS ファイルをフラット補正 FITS ファイルであると認識します。

その後の行程はこれまでと同じです。

3.6 細かい指定をしてみる ~ sirius.param の編集

`sirius.py para` を実行してできる sirius.param というテキストファイルを編集することで、パイプライン処理についての細かい指定ができます。**`sirius.py set`** の実行の前に編集を終えておきます。下は sirius.param の例です。このファイルの読み方は、最初のカラムがキーワード、空白をあけて次が値、さらに空白をあけて最後に # に引き続き簡単な説明です。

```
band all   # [j|h|k|all]
yymmdd 001130   # yymmdd
twflat mflat   # flat name, [j|h|k]+twflat+[.fits]
stop 0   # to stop pipeline after 1:dark+flat 2:sky-subtraction
crremove 0   # cosmic ray removal 0:no 1:min-max method
linear 0   # linearity correction 0:no 1:yes
darkopt 0   # use prepared dark frames? 0:no 1:yes
comsky 0   # combine sky frames? 0:no 1:yes
allcom 1   # combine all the sets with the same object name? 0:no 1:yes
cfitsio 1   # use cfitsio library? 0:no 1:yes
keepd 0   # keep dark.fits files? 0:no 1:yes
keeps 0   # keep sky.fits files? 0:no 1:yes
keepm 0   # keep mask.fits files? 0:no 1:yes
```

sirius.param が生成された時点で、yymmdd と twflat の行以外は、上記の例に限らず上で示した値がデフォルト値として指定されています。この 2 カラム目の値をテキストエディタで書き換える事で細かい指定ができます。

band ： デフォルトでは、`all` 、3 つのバンドを処理するように指定されています。ここを、`j` または `h` または `k` (小文字) に書き換える事で一つのバンドのみを処理するように指定できます。実は、`sirius.py para -band=j` のように sirius.param を生成するときにこの band を指定することもできます。

yymmdd ： これは書き換えることはないはずです。

twflat ： フラット補正 FITS ファイルの名前です。最初の一文字 (バンド名) と最後の .fits の拡張子を抜いた部分の文字列がここに示されます。`sirius.py para` をする前に、作業ディレクトリに同じファイル名で最初の jhk のバンド名だけが違う FITS ファイルを置いておけば、その FITS ファイルがフラット補正 FITS ファイルであると自動で認識されます。あるいは、FITS ファイルがなければ、pyIRSF パッケージが準備しているマスターフラット ([j|h|k]mflat.fits) が使用されます。

stop ： デフォルトの 0 では最後までパイプライン処理を行います。1 にすると dark 引きとフラット割りが終わった時点で終了します。2 にするとスカイバイアス引きが終わった時点で終了します。

crremove ： デフォルトでは特別な宇宙線除去を行いません。1 にすると combine 時に imcombine の min-max reject を行うことで宇宙線除去をします。

linear ： デフォルトではリニアリティ補正を行いません。1 にすると補正を行います。

darkopt ： 同じ日に必要な積分時間の dark を取得していないときがたまにあります。そのようなときは別の日の dark セットから作った [j|h|k]dark30.fits などの dark 補正用 FITS ファイルを作業ディレクトリに持ってきておきます。(sirius.py para 後に !) そして、darkopt の値を 1 にしておきます。

comsky ： デフォルトでは、スカイバイアスのセットのフレームは combine しません。しかし、せっかくなのでその視野がどんな感じか見たい場合にはここの値を 1 にしておけば位置合わせして combine までします。

allcom ： デフォルトでは、obslog で同じ object 名のセットは全て一つのフレームに重ね合わせられます。セット毎に別々に重ね合わせたものが欲しいときにはここの値を 0 にします。

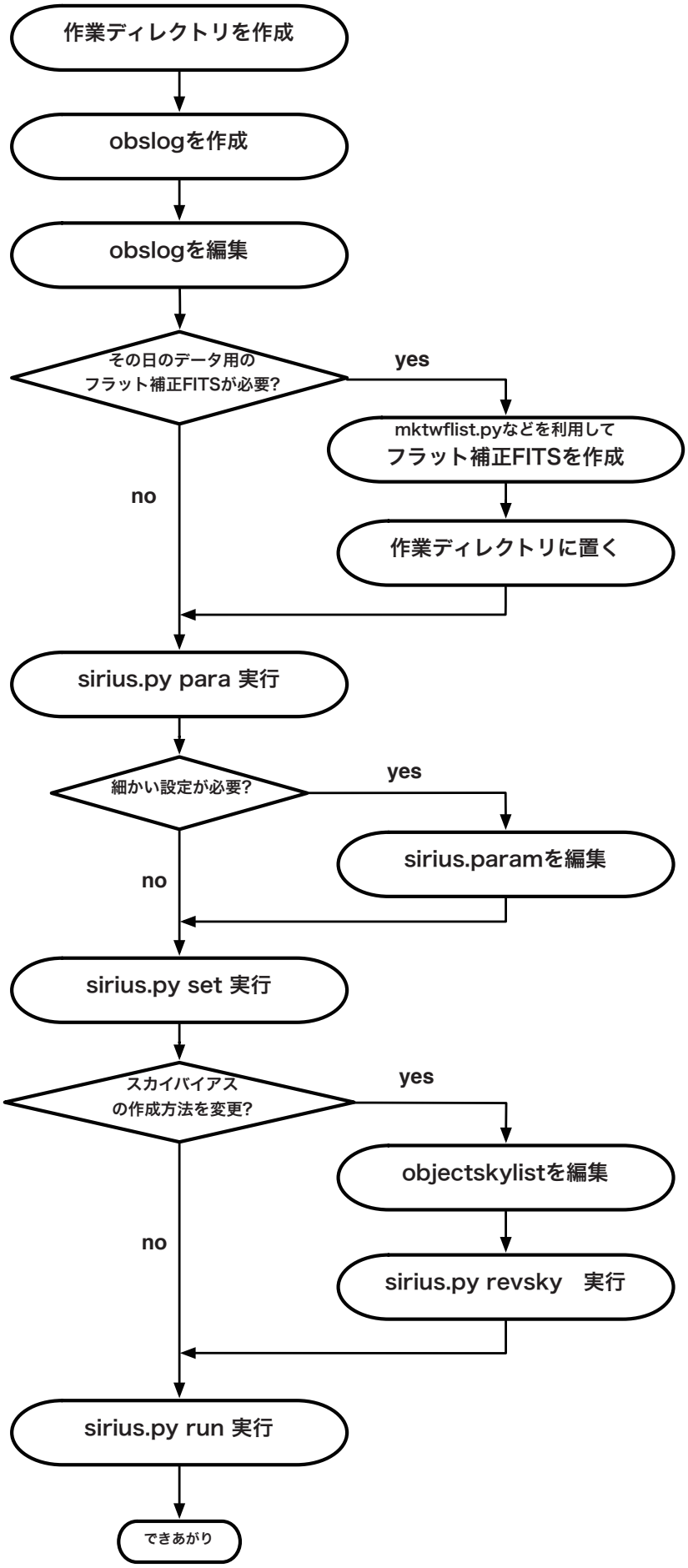
cfitsio ： cfitsio のライブラリを使う事で、生データを解凍しておかなくてもその中からデータを読み取り、処理をすることができます。何らかの事情で cfitsio のライブラリがインストールされていない場合には、解凍済みの生データ FITS ファイルを rawdata のディレクトリに入れておき、ここの値を 0 にしておきます。

keepd ： デフォルトではダーク補正用 FITS ファイル (生データのセットから平均を計算したもの) はパイプライン処理後に消去されます。ここの値を 1 にしておけば消去されません。

keeps ： デフォルトではスカイバイアス補正用 FITS ファイル (生データのセットから平均を計算したもの) はパイプライン処理後に消去されます。ここの値を 1 にしておけば消去されません。

keepm ： デフォルトではバッドピクセルなどをマスクするための FITS ファイルはパイプライン処理後に消去されます。ここの値を 1 にしておけば消去されません。

3.7 処理の流れのまとめ



3.8 ターゲットとスカイの組み合わせのその他のルール

object 名のおしりに sky がついたセットからどのようにスカイバイアスフレームが作られ、使われるかを上で述べました。これ以外に、object 名のおしりに **self** および **map** という文字列をつける事でセルフスカイおよびマッピング観測に対応したスカイバイアスの作成ができます。

[self]

自らのセットからセルフスカイを作成して、それをスカイバイアスとして引き算に使います。そのようにしたいときには、obslog を編集して該当するセットの object 名の後ろに self という文字列を加えます。

[map]

前後のセットおよび自分自身のセットからスカイバイアスを作成します。

```
0051 obj+0+0map 30. 0.0 0.0 2101/10/28 16:55:51.7 2101/10/28 18:55:51.
0052 obj+0+0map 30. 14.6 3.9 2101/10/28 16:56:42.0 2101/10/28 18:56:42
0053 obj+0+0map 30. 8.6 12.3 2101/10/28 16:57:32.5 2101/10/28 18:57:32
0054 obj+0+1map 30. -0.1 14.8 2101/10/28 16:58:22.8 2101/10/28 18:58:2
0055 obj+0+1map 30. -10.7 10.7 2101/10/28 16:59:13.6 2101/10/28 18:59:
0056 obj+0+1map 30. -14.8 0.1 2101/10/28 17:00:05.0 2101/10/28 19:00:0
0057 obj+0+1map 30. -12.3 -8.6 2101/10/28 17:00:55.5 2101/10/28 19:00:
0058 obj+0+2map 30. -3.9 -14.6 2101/10/28 17:01:46.1 2101/10/28 19:01:
0059 obj+0+2map 30. 6.3 -13.7 2101/10/28 17:02:36.5 2101/10/28 19:02:3
0060 obj+0+2map 30. 13.7 -6.3 2101/10/28 17:03:27.1 2101/10/28 19:03:2
```

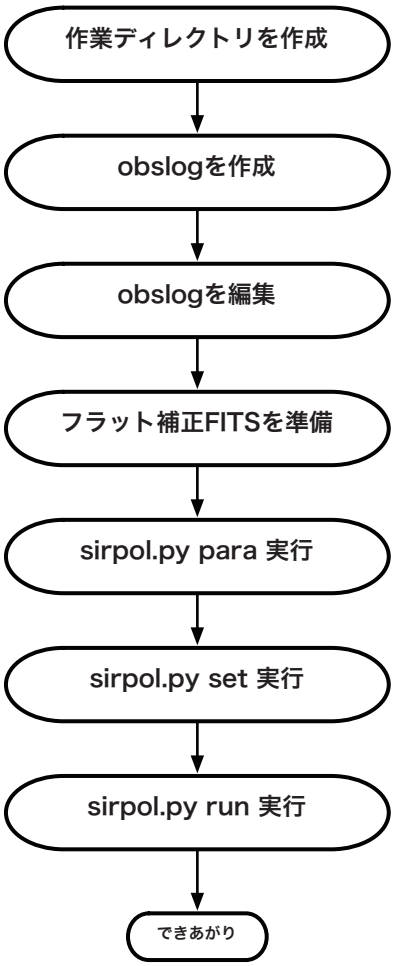
上のように、3 つのセットの obect 名の最後に map をつけます。そうすると、
0051-0053 のセットのスカイバイアスは 0051-0053 および 0054-0057 のフレームから作成されます。次に 0054-0057 のセットのスカイバイアスは 0051-0053 および 0054-0057 および 0058-0060 のフレームから作成されます。最後に、0058-0060 のセットのスカイバイアスは 0054-0057 および 0058-0060 のフレームから作成されます。

4. SIRPOL のデータ処理 ~ 基本編

SIRPOL についてもおおまかな流れは同じです。ただし、SIRPOL のデータ処理には **sirpol.py** を使用します。ここでは 2005 年 12 月 26 日のデータを用いて説明します。

4.1 最も簡単な例

さきほどの SIRIUS の場合と違い、マスターフラットはありません。



実際の例として、051226 の M42 の直線偏光の観測データを処理してみましょう。

(1) rawdata ディレクトリと同じ階層に作業ディレクトリを作成
(2) 作業ディレクトリの中で mklog.py を実行して obslog ファイルを作成
(3) obslog ファイルの編集
までは同じです。この日は M42 のセットを複数回とっていますが、まずは最初のセットだけを取り出して処理してみましょう。obslog で 0291-0370 の M42_n1 と M42_n1sky のセットの行と、1491-1500 の DARK (10 秒) のセット の行のみを残してあとは削除しましょう。
(4) フラット補正 FITS を準備
SIRPOL チームがある時期については、複数日分のトワイライトデータから作ったフラットを用意しています。ここでは **[j|h|k]twfJan06.fits** をもらってきて作業ディレクトリに置きます。2005 年 12 月から 2006 年 2 月の間の 17 夜のデータから作成されたものです。

(5) sirpol.py para 実行

```
$ sirpol.py para
```

これで sirpol.param というファイルができます。sirius.param と同様のものです。しかし、中身が少し違います。後ほど説明します。

(6) sirpol.py set 実行

```
 $ sirpol.py set
```

この後のパイプライン処理に必要なファイルが作成されます。

(7) sirpol.py run 実行

```
$ sirpol.py run
```

パイプライン処理の開始です。SIRIUS のときと同様に、処理のどの段階まで終わったか、各フレームのバックグラウンドレベルや星の fwhm、ディザリングしたフレームの位置合わせの変換の結果などがメッセージとして表示されます。

```
dark frames ...  have been created
dark subtraction and flat division ...  done
making sky bias frames and bad pixel masks ...  done.
done
sky subtraction ...  j h k done

******************
o0.M42_n1
******************
j-band
find stars for geomap/geotran
fnum fwhm ellip star_num thresh    med    stddev
0291  3.5  0.18   30      10.0      96.9  55.1
0292  3.6  0.12   30      10.0      82.8  55.7
0293  3.2  0.09   30      10.0      83.1  55.2
0294  3.4  0.11   30      10.0      80.7  55.7
```

... 途中省略 ...

```
kf0325   0.00   13.7 (0.15)   30.2 (0.11) 186  (220/221)      -0.3  -0.2
kf0326  -0.00   14.1 (0.11)   30.1 (0.10) 152  (220/164)       0.1  -0.3
kf0327  -0.01   30.8 (0.16)   14.4 (0.17) 101  (222/110)       0.4   0.4
kf0328  -0.00   30.8 (0.15)   13.9 (0.09)  96  (222/104)       0.4  -0.1
kf0329  -0.00   30.9 (0.11)   14.2 (0.17) 109  (222/119)       0.5   0.2
kf0330  -0.00   30.9 (0.11)   14.7 (0.13) 109  (222/119)       0.5   0.7
geotran in progress
10% 20% 30% 40% 50% 60% 70% 80% 90% 100% done
making i,q,u, images for each cycle  q  u  i  done
combining
i q u a00 a45 a22 a67
done
cleaning the working directory...
```

(8) 結果を見てみる

処理が終わると作業ディレクトリの中は下のようになっています。SIRIUS のときとほとんど同

```
dithlistlist  htwfJan06.fits  kskybg.list     objectskylist  sirpol.param
ffiles/       jskybg.list     ktwfJan06.fits  obslog
hskybg.list   jtwfJan06.fits  o0.M42_n1/      obslog~
```

じです。一番欲しい結果は **o0.M42_n1** の中にあります。

```
ha00recom.fits  htransummary.txt  jqrecom.fits       ka67recom.fits
ha22recom.fits  hurecom.fits      jquality.txt       kirecom.fits
ha45recom.fits  ja00recom.fits    jtransummary.txt   kqrecom.fits
ha67recom.fits  ja22recom.fits    jurecom.fits       kquality.txt
hirecom.fits    ja45recom.fits    ka00recom.fits     ktransummary.txt
hqrecom.fits    ja67recom.fits    ka22recom.fits     kurecom.fits
hquality.txt    jirecom.fits      ka45recom.fits
```

SIRIUS の時と違って、FITS ファイルがたくさんあります。

[j|h|k]irecom.fits ： Stokes parameter I の画像

[j|h|k]qrecom.fits ： Stokes parameter Q の画像

[j|h|k]urecom.fits ： Stokes parameter U の画像

[j|h|k]a00recom.fits ： 波長板角度 0 度 の画像

[j|h|k]a22recom.fits ： 波長板角度 22.5 度 の画像

[j|h|k]a45recom.fits ： 波長板角度 45 度 の画像

[j|h|k]a67recom.fits ： 波長板角度 67.5 度 の画像

Stokes parameter I, Q, U の画像は波長板角度 0、22.5、45、67.5 度の画像からも作れるのですが、このパイプラインでは観測時の波長板を回す 1 サイクルのフレーム同士で引き算 (Q = I(0) - I(45)、U = I(22.5) - I(67.5)) をした画像をまず作ることで大気透過率の変化の測光値への影響を最小限にし、それらを重ね合わせています。

これら FITS ファイルのヘッダ項目や、もう一つの画像 (各ピクセルが何枚の平均から計算されているか) が含まれていることについては SIRIUS の場合と同じです。

テキストファイルの [j|h|k]quality.txt および [j|h|k]transummary.txt についても SIRIUS の場合と同様です。

作業ディレクトリにある ffiles および [j|h|k]skybg.list についても SIRIUS の場合と同様です。

4.2 同じターゲットを複数のセットにわたって観測した

本当は 9 セットの観測があるのですが、ここでの例題としては 4 セットだけ取り出してみます。

SIRPOL の場合も SIRIUS の場合と同様に操作します。
051226 の M42 の 4 セットのデータを処理してみましょう。

(1) rawdata ディレクトリと同じ階層に新規作業ディレクトリを作成

(2) 作業ディレクトリの中で mklog.py を実行して obslog ファイルを作成

(3) obslog ファイルの編集

0291-0530 の M42 とそのバイアススカイの観測セットおよび 1491-1500 の 10 秒 dark のセットのみを残して、残りは obslog から消去しましょう。
ここで、M42_n1, M42_n2 .. などのターゲットを観測したセットのオブジェクト名を M42 に変えてください。さらに M42_n1sky, M42_n2sky .. などのスカイバイアスを観測したセットについては M42sky に変えてください。SIRIUS のときの「同じターゲットを複数のセットにわたって観測した」の場合の obslog の編集と同様です。

(4) フラット補正 FITS を準備

さきほどと同じものでよいです。この作業ディレクトリにコピーします。

(5) sirpol.py para 実行

(6) sirpol.py set 実行

(7) sirpol.py run 実行

(8) 結果を見てみる

作業ディレクトリのファイルは 4.1 の場合と同じになっています。

複数のターゲットを処理する場合、ターゲットとスカイの組み合わせを変更する場合も SIRIUS の場合 (3.2 および 3.3) と同様に行えばよいです。ただし SIRPOL の場合は sirius.py のかわりに sirpol.py を使うことに注意してください。

4.3 フラット補正 FITS ファイルの準備

SIRPOL チームによって準備されている時期もありますが、そうでない時期のデータを処理するときには自分で作成する必要があります。SIRPOL についても **5. フラット補正 FITS ファイルの作成**の通りにやれば OK です。ただし、波長板をつけてトワイライトフラットを観測しているデータを使って作成してください。

4.4 細かい指定をしてみる ~ sirpol.param の編集

SIRIUS のときと同様に、sirpol.param を編集することでパイプライン処理についての細かい指定ができます。

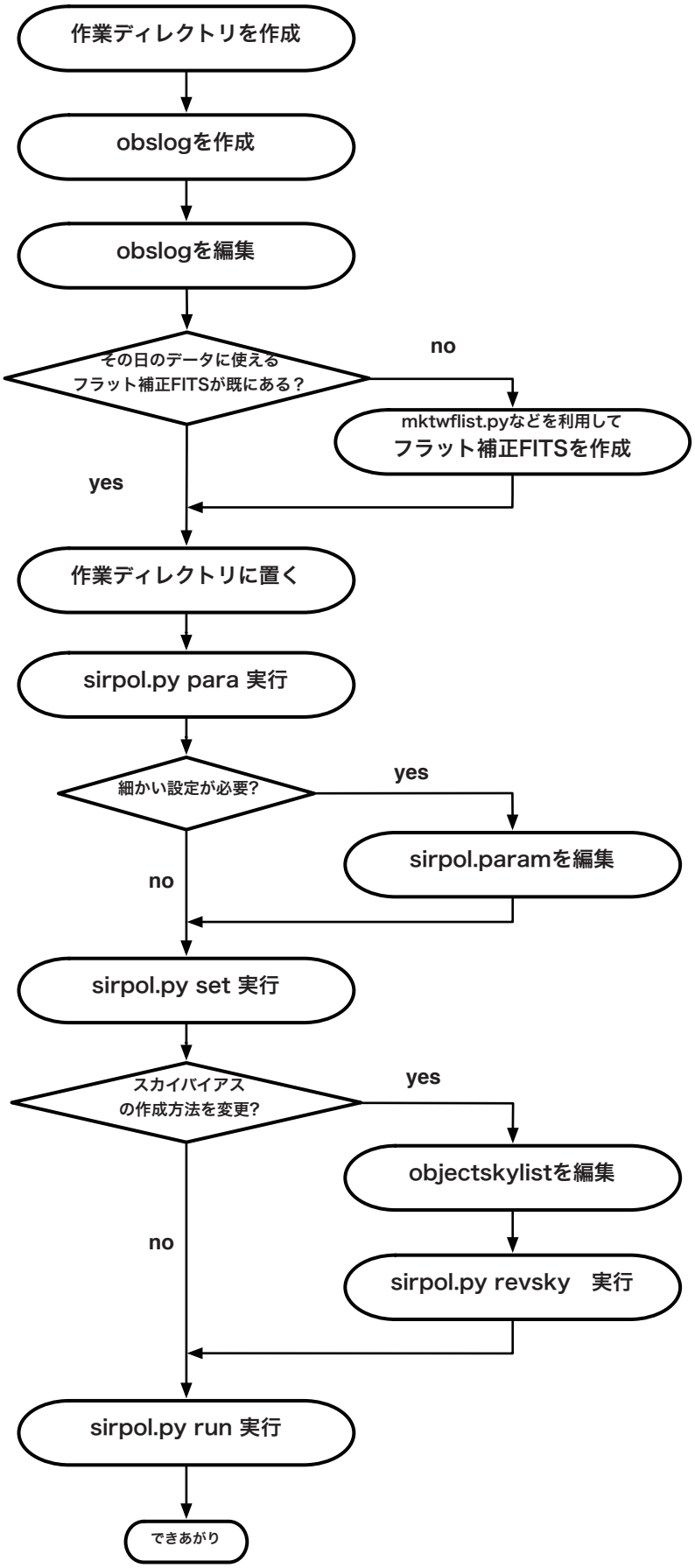
```
band all   # [j|h|k|all]
yymmdd 051226   # yymmdd
twflat twfJan06   # flat name, [j|h|k]+twflat+[.fits]
stop 0   # to stop pipeline after 1:dark+flat 2:sky-subtraction
crremove 0   # cosmic ray removal 0:no 1::min-max
linear 0   # linearity correction 0:no 1:yes
darkopt 0   # use prepared dark frames? 0:no 1:yes
comsky 0   # combine sky frames? 0:no 1:yes
allcom 1   # combine all the sets with the same object name? 0:no 1:yes
polmode 1 # 1:LP, 2:CP2, 3:CP4
cfitsio 1   # use cfitsio library? 0:no 1:yes
fiim 1 # make I image? 0:no 1:yes
fqim 1 # make Q image? 0:no 1:yes
fuim 1 # make U image? 0:no 1:yes
fvim 0 # make V image? 0:no 1:yes
fpim 0 # make PI image? 0:no 1:yes
f00im 1 # make 0deg image? 0:no 1:yes
f22im 1 # make 22.5deg image? 0:no 1:yes
f45im 1 # make 45deg image? 0:no 1:yes
f67im 1 # make 67.5deg image? 0:no 1:yes
f90im 0 # make 90deg image? 0:no 1:yes
f135im 0 # make 135deg image? 0:no 1:yes
keepd 0   # keep dark.fits files? 0:no 1:yes
keeps 0   # keep sky.fits files? 0:no 1:yes
keepm 0   # keep mask.fits files? 0:no 1:yes
```

最初から allcom までと、cfitsio および最後の三行については sirius.param と同じです。

polmode ： 1 直線偏光か、2 円偏光二点法か、3 円偏光四点法かどの観測データを処理するのかを指定します。デフォルトは 1 の直線偏光です。このパイプラインでは直線偏光と円偏光のデータを同時に処理する事はできません。

fiim ： 最終的に重ねたIイメージを作るか？ デフォルトは 1 (0 は no、1 は yes 以下同じ)
fqim ： 最終的に重ねた Q イメージを作るか？ デフォルトは 1
fuim ： 最終的に重ねた U イメージを作るか？ デフォルトは 1
fvim ： 最終的に重ねた V イメージを作るか？ デフォルトは 0
fpim ： 最終的に重ねた PI イメージを作るか？ デフォルトは 0
f00im ： 最終的に重ねた波長板 0 度イメージを作るか？ デフォルトは 1
f22im ： 最終的に重ねた波長板 22.5 度イメージを作るか？ デフォルトは 1
f45im ： 最終的に重ねた波長板 45 度イメージを作るか？ デフォルトは 1
f67im ： 最終的に重ねた波長板 67.5 度イメージを作るか？ デフォルトは 1
f90im ： 最終的に重ねた波長板 90 度イメージを作るか？ デフォルトは 0
f135im ： 最終的に重ねた波長板 135 度イメージを作るか？ デフォルトは 0

4.5 処理の流れのまとめ



5. フラット補正 FITS ファイルの作成

5.1 マスターフラット

フラット補正データ作成のために、トワイライトの空の明るさが変化していく様子を観測します。明るいスカイから暗いスカイの差をとったものが一様光源を観測したものと等価であると仮定して、ピクセル間の感度ムラおよび光学系透過率のムラを補正するフラット補正 FITS ファイルを作成します。明るいスカイから暗いスカイの差をとった後に全ピクセル値のメジアンで規格化したものをなるべく多く集めて、それらメジアンのイメージをフラット補正 FITS ファイルとします。

ここで、一様光源を仮定したのですが、実際には (おそらく)SIRIUS 内部の迷光の影響で一様光源とはなっていません。その非一様成分は視野の端のほうで 0.05 等かそれ以上のズレを引き起こします。(あるいは、要求する測光精度が 0.05 等程度であれば、上記のようにして作成しただけで十分なフラット補正 FITS ファイルができあがっていることになります)

この非一様成分を補正するために、2000 年 4 月から 2004 年 4 月の観測データを用いて、SIRIUS の測光値と 2MASS の測光値の差の SIRIUS アレイ上の (x, y) の位置による依存性を調べました。その依存性から、上記のようにして作成したフラット補正データを補正するデータを作成しました。そのようにして作成されたのが pyIRSF パッケージに入っているマスターフラットです。このマスターフラットによる測光誤差は 0.02-0.03 等級以内であると見積もっています。

詳細は http://www.z.phys.nagoya-u.ac.jp/~nakajima/yas/repository/corrected-flat.pdf

マスターフラットは、少なくとも 2000 年から 2004 年のデータ処理に使う分には問題ありません。これまで調査した分では、2008 年までのデータについても 1% 以内の精度で問題ないことがわかっています。その根拠は以下です。この (おそらく) 迷光による非一様成分は、SIRIUS に固有のものであると考えています。従って、経年変化しないと仮定しています。2000 年から 2004 年のトワイライトのデータから作成した「補正前」のフラット補正 FITS ファイルと 2008 年までのものとで、比をとると 1% 以内で一致するからです。

5.1 独自のフラット補正 FITS ファイル

マスターフラットでは適切ではないと考える場合には、そのデータが観測された日に近い任意の期間のトワイライトフラットのデータを用いて、独自にフラット補正 FITS ファイルを作成します。なるべく多くの日のトワイライトフラットのデータを組み合わせるのをおすすめします。理由は、(1) フラット補正特性は少なくとも数ヶ月くらいでは変化しない、(2) 一時的に雲が通った場合などの悪いデータを除外しやすい、(3)S/N を上げる、が挙げられます。

フラット補正 FITS ファイルを作成するためには、pyIRSF パッケージにある **mktwflist.py**、**twfcom.py**、**corrflat.py** およびオプションとして **twflatview.py** を使用します。

5.2 ファイルとディレクトリの準備

生データの入っている日付ディレクトリと同じ階層に作業ディレクトリを作ります。

ここでは作業ディレクトリの名前を仮に wrk としましたが、もちろん他と重ならなければ何でもよいです。

```
$ mkdir wrk
$ ls
 030219/  030211/  030214/ wrk/
```

それぞれの日付ディレクトリの中には SIRIUS 標準ディレクトリ構造にしたがって、rawdata というディレクトリがあるはずです。その中には生データがあります。これら生データは通常の fits ファイルおよび、fpack 圧縮ファイル .fits.fz または imcopy 圧縮ファイル .fits.ic のいずれでもかまいません。つまり圧縮ファイルは解凍しなくてもよいです。

ここで作った作業ディレクトリの中に移動して、フラットデータを利用する日付を指定するファイルを作ります。以下の例のように日付を並べます。

この名前も dlist でなくてもかまいません。この後作成されるファイルの名前、jflatlist, hflatlist, kflatlist などと重ならなければ OK です。

```
$ more dlist
030129
030211
030214
```

ここに書いた日付のすべてのフラットデータから、各バンド毎にひとつのフラット補正用の FITS ファイルが作成されます。少なくとも数ヶ月程度ではアレイの特性や光学系の透過率なんて変化しないので、処理したいデータの日付付近のなるべく多くの日付のものを重ね合わせるのがよいです。

「どの日にトワイライトフラット観測したっけ？」は心配しなくてよいです。指定された日付のディレクトリに FITS ヘッダの Keyword OBJECT が twflat であるファイルがない場合には、その日付はスキップされるだけです。

ナローバンド用のフラットを観測した場合、NB14twflat のような OBJECT 名をつけますが、そのときには後述のようにキーワードを NB14twflat などと指定することで、そのフラットのみを取り出すことができます。

5.3 プログラム実行 ~ 基本 ~

(1) 作業ディレクトリの中で mktwflist.py を実行します。

```
$ mktwflist.py dlist
```

最初の引数で日付指定ファイルを指定します。

実行すると、処理中の日付が次々表示されます。

日付指定ファイルで指定した日付に twilight flat の観測がなければ

```
No flat frames taken on 030317
```

のように画面に表示されます。そのまま続けて問題ありません。処理の最後には

```
a total of 62 pairs are selected for j
a total of 59 pairs are selected for h
a total of 47 pairs are selected for k
Done
```

のように、いくつのペアが選ばれたかの結果が表示されます

この結果、 [j|h|k]flatlist.0, [j|h|k]flatlist, [j|h|k]backround.txt, mktwflist.param というファイルができます。

(2) 次に twfcom.py を実行します。

```
$ twfcom.py flat0302
```

引数として出来上がるファイルの名前を指定します。この例の場合、[j|h|k]flat0302.fits という名前のファイルが出来上がります。

mktwflist.py の結果の [j|h|k]flatlist の各行で指定されたペアの引き算画像をそのメジアンで規格化したものを集めて、median でコンバインしたものが最終的なフラットフレームになります。出来上がった FITS ファイルのヘッダには NCOMBINE というキーワードがあり、何枚重ね合わされたかの数字が入ってます。

corrflat.py で行っている、処理や採用している仮定が怪しいと思う人は、これを実行せずに twfcom.py の結果ファイルをフラット補正 FITS ファイルとするのもアリでしょう。その場合、非一様成分によって視野の端のほうでは測光精度が悪くなる事を留意しておいてください。

(3) corrflat.py を実行します。

```
$ corrflat.py flat0302 nflat0302
```

引数として、最初に twfcom.py で作成した FITS ファイルの名前を与え、二番目に新しくできる FITS ファイルの名前を与えます。この例の場合、[j|h|k]flat0302.fits のファイルに補正をかけて新しく [j|h|k]nflat0302.fits という名前のファイルを作成します。

この処理では、迷光による非一様成分の補正を行います。ここでは、マスターフラットを作成した時期と比べて、その非一様成分は変化していないと仮定しています。

マスターフラットのピクセル値分布を mflat(x, y) として、それを作成するときに最初の段階で作成したフラット補正 FITS ファイル (非一様成分の補正なし) のピクセル値分布を cflat(x,y) とします。 先ほど twfcom.py で作成したフラット補正 FITS ファイルのピクセル値分布を iflat(x, y) とします。この処理で得られる非一様成分補正済みフラット補正 FITS ファイルのピクセル値分布を nflat(x, y) とすると、この処理では

```
nflat(x, y) = iflat(x, y) / cflat(x, y) * mflat(x, y)
```

という計算をしています。

5.4 プログラム実行 ~ 詳細な指定と出力の見方 ~

(1) mktwflist.py 実行時にオプションの引数を与えて細かい指定を与えることもできます。

```
$ mktwflist.py dlist -noexpgap
```

段差の小さいペアの選出については後述。

とすれば mktwflist.py は段差の小さいペアの選出をせずにプログラムを終了します。

```
$ mktwflist.py dlist -skydiff=1000
```

デフォルトではバックグラウンドレベルの差が 2000 カウント以上のものを選び出しています。

-skydiff=[float] のオプションではその差を指定することができます。

これらの他のオプションとして

-maxgood=[float] (例 : **-maxgood=6500**)

デフォルトではバックグラウンドレベルが 6000 未満のもののみを使います。そのしきい値を指定することができます。

-keyword=[string] (例 : **-keyword=NB14twflat**)

デフォルトでは FITS ヘッダの keywrod OBJECT が twflat のものの中からペアを選び出します。その keyword を別のものに変えることができます。

-band=[string] (例 : **-band=j -band=h,k**)

あるバンドのもののみを作りたい場合にそのバンドを指定します。カンマで区切ることで二つのバンドを指定することもできます。

何も引数を与えずに

```
$ mktwflist.py
```

とだけしてやると、オプション引数の簡単な一覧が表示されます。

引数に help の文字が含まれている場合にも表示されます。

(2) mktwflist.py 実行中の警告メッセージ

mktwflist.py の処理途中で、バックグラウンドレベルの変化を調べています。単調に明るくなるあるいは暗くなるという傾向から外れた場合には視野の中に雲が入ったのかもしれません。J バンドのフレームでそのようなことがあった場合には

```
background level unstable at 030313 0010
```

のように警告メッセージが出ます。mktwflist.py の処理後に jbackground.txt というファイルが同じディレクトリにあるので、その前後のファイルをチェックしてみましょう。unstable だと指摘があったフレームの行の右に * で印があります。H と K バンドのバックグラウンドも [h|k]background.txt に記録されます。H と K の場合にはバックグラウンドレベルの変化は熱や夜光の状態にも依存するので、トワイライトフラット観測時に明るさが単調増加 / 減少するとは限りません。ですので、画面にいちいち警告メッセージを表示しません。ですが、[h|k]background.txt には * で印をつけるようにしています。

警告がなかった場合には mktwfwarning.txt は作成されません。

この警告メッセージは、同じディレクトリの mktwfwartning.txt にも記録されます。

(3) mktwflist.py 実行後にできるファイル

[j|h|k]flatlist, [j|h|k]flatlist.0

この [j|h|k]flatlist が mktwflist.py の出力で一番重要なファイルです。

下は jflatlist の例です。

いったいどんなペアが選び出されたのか、全くのブラックボックスでは気持ちの悪い人のためにそれぞれのペアのバックグラウンド値も見れるようにしておきました。

```
$ more jflatlist
030129 0025 0036 3310.0 1254.0
030129 0026 0040 3001.0 928.0
030129 0027 0044 2724.0 720.0
030129 0914 0909 4712.0 2468.0
030129 0913 0907 4157.0 1912.0
030129 0912 0905 3651.0 1465.0
.... 以下省略 ....
```

このリストはどう読むかというと、最初の行の場合「(これは "jflatlist" なので)J バンドの 030129 の日付の 0025 番と 0036 番フレームのペアの差をとる。それぞれのバックグラウンドのメジアンは 3310.0 と 1254.0。」こういうペアが何行にもわたって記載されています。

jflatlist.0 も同様の内容です。.0 とそうでないほうの違いは、[j|h|k]flatlist.0 のペアの中から、reset anomallly 起源の中央の段差が、そのペアの引き算の結果 1% 以下のもののみを選び出したものが [j|h|k]flatlist　となっています。

この「段差の小さいもののみの選出」のアルゴリズムについては 5.5 解説を参照。

引数で **-noexgap** を指定すると、その選び出しをしません。[j|h|k]flatlist.0 は作成されずに、[j|h|k]flatlist のみが作成されます。

mktwflist.param には設定したパラメータが記録されます。何も引数で指定していなければデフォルトの値が入っています。

[j|h|k]background.txt

先述のように、各バンドのフラット観測フレームのバックグラウンドレベル (メジアン) が記

録されています。バックグラウンドが暗い方から明るい方に変化するような順になっています。つまり夕方のフラット観測フレームセットの場合、フレーム番号が減る順になっています。
行の右端に * があるフレームは上の行のフレームと比べてバックグラウンドレベルが小さくなっているものです。

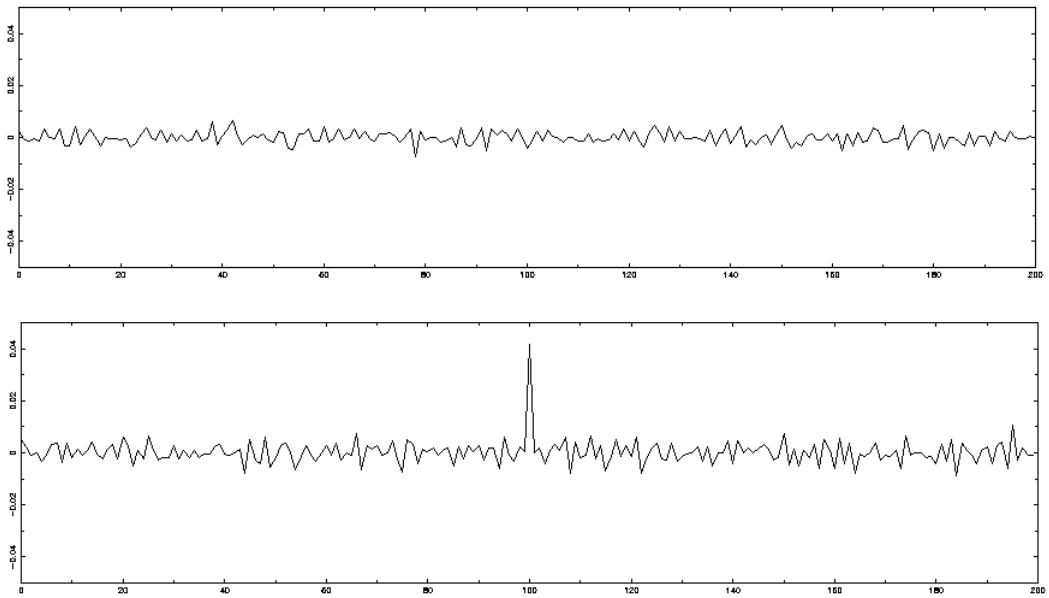
(4) twfcom.py 実行時のオプション
`-band=[string]`（例：**`-band=j`**　**`-band=h,k`**）
あるバンドのもののみを作りたい場合にそのバンドを指定します。カンマで区切ることで二つのバンドを指定することもできます。

(5) 必要であれば、jflatlist, hflatlist, kflatlist をテキストエディタで編集します。
このペアは不要だというのものがあれば、その行を削除あるいは # を先頭につけてコメントアウトします。すると次の twfcom.py ではそれは無視されます。

5.5 解説 ~ 段差の小さいのだけを選び出す ~

この「段差」は HAWAII アレイの reset anomally という特性によって生じます。reset anomally によるパターンが安定していれば、明るいスカイから暗いスカイを引いた差のイメージには「段差」が残りません。時々不安定になるとこの段差が見られます。

ペアの引き算したイメージに対して、413-column から 612-column までについて、左隣の column との差について 1-row から 1024-row までの間のメジアンを計算します。これでメジアン値が 200 個そろいます。この 200 個のメジアンと標準偏差を計算します。メジアンはほとんど 0 の値です。 511-column と 512-column の差が 1% より小さいか 3-sigma より小さい場合には、段差が小さいと判断します。



図は、横軸に column number-412 (200 点ある) をとり、縦軸にバックグラウンドのメジアンで規格化した「左隣の column との差」をとりました。上図の例では中央の段差はみられません。下図の例では中央の段差がはっきりと見られます。

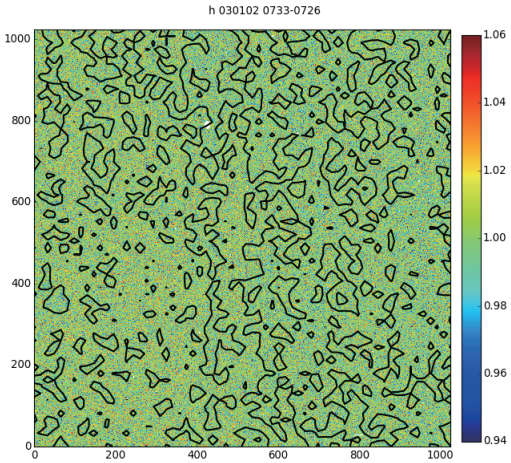
5.6 目で見て判断 -- twflatview.py

オプションの twflatview.py を使うことで、振る舞いの変なペアの選出が容易にできます。
各ペアの引き算イメージを最終的なメジアンフラットで割り算したものを次々と表示します。
(これを使うには python のモジュール matplotlib が必要です。)

上記の mktwflist.py で得られたリストを twflatview.py にかけてやります

```
$ twflatview.py hflatlist hflat.fits
```

するとリストの最初の行のペアから順に「ペアの引き算」/「最終的なメジアンフラット」の図がコントア付きで表示されます。下に例を示します。

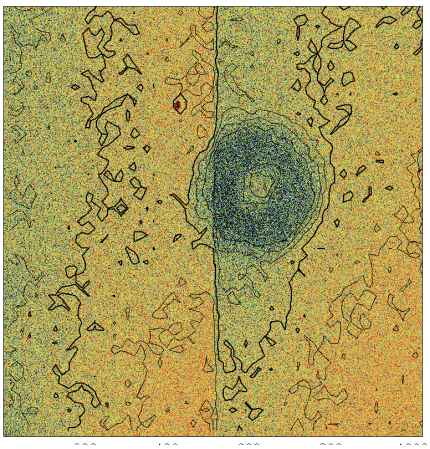



```
next :
```

と出るのでリターンキーを押すことで次々と出てきます。画面中央上に h 030102 0733-0726 といったようにどのペアかが表示されます。

16x16 ピクセルでメジアンをとってスムージングをかけたコントアが表示されます。1.00 のコントアは太い線で表示され、それ以外のレベルは 0.01(=1%) おきに細いコントアで表示されます。通常は 1.00 のコントアしか見えません (上の図)。

次の図では、はっきりとしたパターンで 2% かそれ以上のレベルが見えています。これらを削除したい場合にはフレーム番号を確認した上で、hflatlist などのリストの該当する行を削除または # でコメントアウトしてから twfcom.py で flat を作成します。



6. 測光と位置測定およびその較正

6.1 sirphot.py

sirius.py や sirpol.py の結果の重ね合わせた FITS ファイルを測光するためのアプリです。アパーチャー測光をします。

[使い方]

```
$ sirphot.py jrecom.fits
```

のように **sirphot.py [FITS ファイル名]** としてやります。その結果、**[バンド名]sirphot.txt** というテキストファイルが作成されます。 この場合なら jsirphot.txt です。

```
536.2 12.9 15.044 0.012 0 0 0 3.494 0.13
505.3 24.4 12.614 0.001 0 0 0 3.610 0.12
650.6 24.1 15.244 0.014 0 0 0 3.720 0.49
911.0 27.9 15.046 0.010 0 0 0 3.826 0.14
792.5 29.9 16.163 0.029 0 0 0 INDEF INDEF
779.0 33.3 15.434 0.015 0 0 0 INDEF INDEF
820.6 37.1 15.316 0.011 0 0 0 3.704 0.07
425.8 38.0 15.827 0.017 0 0 0 3.288 0.39
787.7 42.1 14.827 0.007 0 0 0 INDEF INDEF
944.5 46.6 15.446 0.013 0 0 0 INDEF INDEF
666.9 50.5 15.621 0.014 0 0 0 3.610 0.34
128.1 50.8 15.631 0.011 0 0 0 3.714 0.21
919.9 55.3 15.656 0.013 0 0 0 3.754 0.50
955.2 55.7 13.672 0.002 0 0 0 3.724 0.17
348.3 56.7 15.485 0.010 0 0 0 3.802 0.26
608.3 59.4 15.915 0.021 0 0 0 3.828 0.43
```

各行にそれぞれの星の情報が記載されています。各カラムは、1 と 2 が星の重心の位置の x, y 座標 (pixel)、3 と 4 には等級とそのエラー、5、6、7 は iraf.apphot の cier, sier, pier、 8 と 9 は fwhm(pixel) と ellipticity で す。1~7 は iraf.apphot の 測 定 結 果 で、8 と 9 は iraf.psfmeasure の測定結果です。この段階では、等級はまだ機械等級です。つまり較正されていません。iraf.daofind と iraf.apphot と iraf.psfmeasure をしただけです。

sirphot.py でも細かい処理の指定をすることができます。

-ifwhm ： 星の fwhm があまりにも大きい (fwhm > 6) ときにはだいたいのサイズを与えておく方がよいでしょう。 最初の daofind で使う fwhm の値です。デフォルトでは 3 です。

-thresh ： 星の検出のときのしきい値です。(iraf.apphot.datapars.threshold) デフォルトは 10 です。

-aprad ： 星の測光の半径です。(iraf.apphot.photpars.apertures) デフォルトでは iraf.psfmesure で測定した fwhm(視野の星のメジアン値) です。

結果の 8、9 カラムの fwhm と ellipticity には INDEF の値が入る事があります。これは、iraf.psfmeasure の測定の結果の星の中心座標が、iraf.apphot の重心の座標と比べて fwhm(視野の星のメジアン値) 以上ずれる場合です。fwhm の測定がちゃんとできなかったということです。星じゃないものを測定している可能性があります。

6.2 sirwcs.py

sirius.py や sirpol.py の結果の重ね合わせた FITS ファイルを (1) アパーチャー測光および 2mass カタログと比較して (2) 星の (x, y) 座標を (α、δ) に変換、さらに (3) 測光値を較正、(4) FITS ヘッダに wcs 情報を書き込む、ということをします。

[使い方]

```
$  sirwcs.py  jrecom.fits
```

のように **sirwcs.py [FITS ファイル名]** としてやります。その結果、**[バンド名]sirphot.txt、[バンド名]sirphotwcs.txt** というテキストファイルが作成されます。
インターネットに繋がっている必要があります。
なお、sirwcs.py は sirphot.py をその中で行ってから 2mass カタログとの比較を行い、位置と測光値の較正を行いますので、sirphot,.py と sirwcs.py を両方するのは無駄です。
[バンド名]sirphot.txt については先述の sirphot.py の結果のものと同じです。較正はされていません。[バンド名]sirphotwcs.txt については較正されています。

```
536.2 12.9 5:38:40.17 -69:09:33.3 14.744 0.012 0 0 0 3.494 0.13
505.3 24.4 5:38:42.84 -69:09:28.7 12.314 0.001 0 0 0 3.610 0.12
650.6 24.1 5:38:30.50 -69:09:26.0 14.944 0.014 0 0 0 3.720 0.49
911.0 27.9 5:38:08.41 -69:09:19.0 14.746 0.010 0 0 0 3.826 0.14
792.5 29.9 5:38:18.48 -69:09:20.5 15.863 0.029 0 0 0 INDEF INDEF
779.0 33.3 5:38:19.64 -69:09:19.2 15.134 0.015 0 0 0 INDEF INDEF
820.6 37.1 5:38:16.12 -69:09:16.7 15.016 0.011 0 0 0 3.704 0.07
425.8 38.0 5:38:49.63 -69:09:24.0 15.527 0.017 0 0 0 3.288 0.39
787.7 42.1 5:38:18.93 -69:09:15.1 14.527 0.007 0 0 0 INDEF INDEF
944.5 46.6 5:38:05.64 -69:09:09.9 15.146 0.013 0 0 0 INDEF INDEF
666.9 50.5 5:38:29.21 -69:09:13.7 15.321 0.014 0 0 0 3.610 0.34
128.1 50.8 5:39:14.95 -69:09:23.8 15.331 0.011 0 0 0 3.714 0.21
919.9 55.3 5:38:07.76 -69:09:06.4 15.356 0.013 0 0 0 3.754 0.50
955.2 55.7 5:38:04.77 -69:09:05.5 13.372 0.002 0 0 0 3.724 0.17
348.3 56.7 5:38:56.28 -69:09:17.1 15.185 0.010 0 0 0 3.802 0.26
608.3 59.4 5:38:34.22 -69:09:10.8 15.615 0.021 0 0 0 3.828 0.43
```

このファイルには、3,4 カラムに赤経赤緯の値が入っています。1, 2 および 5 以降の並びは先ほどの [バンド名]sirphot.txt と同様です。ただし、このファイルでは等級は較正されています。ただし、システム変換はされていません。ここで測光した星と 2mass カタログの星とをマッチさせ、等級差のメジアンを計算してその値で較正しただけです。

sirwcs.py でも sirphot.py と同様の細かい処理の指定をすることができます。
-fwhm, -thresh, -aprad については同じです。

-jhk ：　j, h, k 全部を一気に処理します。sirwcs.py jrecom.fits -jhk　のように最初の引数には j,h,k のどれでもいいからファイル名を指定します。

-noget2mass ：　もうその視野の 2mass カタログをとってきている場合 (2mass.out がある) には、これをつけるとネッ経由で再びとることはありません。-fwhm などのパラメータを変えてやり直すときに使います。

7. 位置合わせと重ね合わせだけを行う

7.1 imgrecom.py

sirius.py の結果の ffiles のディレクトリの中には、ダーク引き、フラット補正、スカイバイアス引きが済んだフレームがあります。これらの位置合わせと重ね合わせだけ行いたいときに、imgrecom.py を使います。
sirius.py 実行時に位置合わせがうまく行かなかったのでやりなおす、他の日の同じ視野の観測を一緒に重ね合わせたいという場合に使います。

[ある一日だけのフレームを重ねる]
重ねたいフレームが入っている ffiles ディレクトリがあるディレクトリに作業ディレクトリを作ります。

```
$ ls
ffiles/   wrk/
```

まず、その作業ディレクトリ内で **imgrecom.py init [band]** とします。

何も引数をつけずに imgrecom.py だけうつと説明があらわれます

```
$  imgrecom.py init  j
```

のようにバンド毎に行います。どれか一つのバンドだけやり直すということが多いのでこのように分けています。
すると、作業ディレクトリ内に **[band]imgrecom.list** というテキストファイルができます。上の例ですと、jimgrecom.py ですね。 このファイルの中には ffiles の中の全てのフレームの番号や初期オフセットの情報などが、フレームについて一行づつ記載されています。

```
0.ffiles 0189 ngc2071n1 0.0 0.0 30.0
0.ffiles 0190 ngc2071n1 19.4 5.2 30.0
0.ffiles 0191 ngc2071n1 10.0 17.4 30.0
0.ffiles 0192 ngc2071n1 -5.2 19.4 30.0
0.ffiles 0193 ngc2071n1 -17.4 10.0 30.0
0.ffiles 0194 ngc2071n1 -19.4 5.2 30.0
0.ffiles 0195 ngc2071n1 -10.0 -17.4 30.0
0.ffiles 0196 ngc2071n1 5.2 -19.4 30.0
0.ffiles 0197 ngc2071n1 17.4 -10.0 30.0
0.ffiles 0198 ngc2071n1sky 0.0 0.0 30.0
0.ffiles 0199 ngc2071n1sky 19.4 5.2 30.0
0.ffiles 0200 ngc2071n1sky 10.0 17.4 30.0
0.ffiles 0201 ngc2071n1sky -5.2 19.4 30.0
0.ffiles 0202 ngc2071n1sky -17.4 10.0 30.0
0.ffiles 0203 ngc2071n1sky -19.4 5.2 30.0
0.ffiles 0204 ngc2071n1sky -10.0 -17.4 30.0
0.ffiles 0205 ngc2071n1sky 5.2 -19.4 30.0
0.ffiles 0206 ngc2071n1sky 17.4 -10.0 30.0
0.ffiles 0207 ngc2071n2 0.0 0.0 30.0
0.ffiles 0208 ngc2071n2 19.4 5.2 30.0
0.ffiles 0209 ngc2071n2 10.0 17.4 30.0
```

次に、重ねたいフレームだけの行を残します。ここでは ngc2071n1sky なんてのは削除します。ngc2071n1 と ngc2071n2 が同じ視野であれば、ここでのオブジェクト名の違いはそのままほっておいてください。
次に、**imgrecom.py recom [band]** とします。

```
$  imgrecom.py recom  j
```

あとは、sirius.py のときと同様に途中結果などが表示され、最後に重ね合わせた FITS ファイルなどが作成されます。

```
$ ls
jquality.txt    jrecom.fits    jtransummary.txt
```

これらの見方は sirius.py の結果のものと同じです。

imgrecom.py recom 実行時のオプション（ 主なもの)

-search ： 星の数が少ない視野では、iraf.xyxymatch を使いません。基準となるフレームでのある星と、別のフレームの星とを同定するときには、基準のフレームでのその星の x, y から半径 が この -search= であたえられた値 (pixel) 以内のものであれば、その別フレームの星は同じ星であると見なせ、のようにします。ディザリング半径よりも大きくとります。しかし、視野内の平均的な隣の星までの距離よりも十分小さくとります。デフォルトは 20 です。

-ufwhm ： この値 (pixle) よりも大きな fwhm のフレームは重ね合わせません。あまりにもシーイングサイズが大きいものを除外するために使います。デフォルトは 20 です (ないも同然です)。

-urms ： iraf.geomap でのフレームの位置合わせの rms の上限値。(pixel) デフォルトは 1。位置合わせの精度が悪いものを除外するときに使います。

-lstarnum ： 位置合わせの計算に使われる星の数の下限値。デフォルトは 1。

-ltheta ： 視野の真上が北でないときには、位置合わせは失敗します。真上が北から反時計回りに何度回転しているかを与えます。

-ifwhm ： 星の fwhm があまりにも大きい (fwhm > 6) ときにはだいたいのサイズを与えておく方がよいでしょう。 最初の daofind で使う fwhm の値です。デフォルトでは 3.5 です。

-reject ： imcombine の reject です。デフォルトは sigclip です。 宇宙線が気になるときは minmax を指定しましょう。

[複数の日のフレームを重ねる]
120102 と 120103 の処理を sirius.py で行った後に、次のようにします。
作業ディレクトリを作成します。
そのディレクトリに、120102 の作業ディレクトリ内の ffiles ディレクトリを 120102.ffiles という名前にしてコピーし、120103 の作業ディレクトリ内の ffiles ディレクトリを 120103.ffiles という名前にしてコピーします。
そのディレクトリにさらに作業ディレクトリを作成します。

```
$ ls
120102.ffiles/    120103.ffiles/    mywrk/
```

その作業ディレクトリ (上の例では mywrk) 内で **imgrecom.py init [バンド]** としてやります。
次に先ほどと同様に、[バンド]imgrecom.list を編集して、重ねたいフレームだけの行を残します。
imgrecom.py recom [バンド]
としてやれば、さきほどと同様に重ね合わせ処理が行われ、結果ファイルができあがります。

7.2 polrecom.py

sirpol.py の結果の ffiles について重ね合わせる場合、polrecom.py を使用します。上記の imgrecom.py の記述中の imgrecom.py -> polrecom.py に読み替え、sirius.py -> sirpol.py に読み替えるだけです。