

# 東北大屋上観測データ処理用スクリプト集 TOKRED 使用マニュアル

作成: 馬渡健

mawatari@astr.tohoku.ac.jp

改訂: 2014年10月

イントロダクション

# TOKRED 使用のための環境・条件

- iraf, sh, SExtractor, c 言語, gnuplot
- iraf に外部パッケージとして stddas が必要 (imfillタスク使用)
- 東北大青葉山キャンパス物理 A 棟屋上 50cm 望遠鏡+ ApogeeAlta U9000 CCD カメラによる撮像観測を想定
- 観測が理想的に行われたこと (良質な生データ) を想定

※ 本資料内で” % ” から始まる行はコマンド端末上での作業を、” ecl> ” から始まる行は iraf での作業を表す。

# TOKRED 内容物

マニュアル & 装置情報などが載ってるポスター (telescope.pdf)

- 1、mkdark.cl ... dark または bias 作成
- 2、mkflat.cl ... flat 作成
- 3、flatfield.cl ... object フレームの dark 引き & flat 割り
- 4、subsky-simple.cl ... 単純な sky 引き
- 5、obj-detection.sh ... 複数画像への天体検出とカタログ作成
- 6、position-match.cl ... 画像の位置合わせ (xyxymatch, geomap, geotran)
- 7、final\_imcomb.cl ... 位置合わせされた画像のスタッキング
- 8、mknoiseimage.cl ... 画像の天体部分をノイズで埋めて仮想 sky 画像作成
- 9、estbackerr.cl ... 画像のバックグラウンドノイズのゆらぎ ( $1\sigma$ ) を測定する
- 10、est\_seeing.cl ... 複数の星を使って、画像のPSFを求める

( おまけ ) estTIME.cl ... 画像の時刻を調べる

( おまけ ) namechange.sh ... 画像の名称変更 (あまり使い勝手よくない)

( おまけ ) cp-file.sh ... あるリストに書かれたファイルを別のディレクトリにコピー

(なんとなく付けといた) transit\_special/calc-flux\_ratio.sh, time-stack.c ...

OTS(屋上トランジット作戦)で使えるかもしれないプログラム



# TOKRED 内容物

TOKRED\_sampledata/・・・サンプルデータ:2012/6/22に東北大屋上51cm望遠鏡で取得

ターゲットはM51

B,V,R,Ha,[OIII],[SII]バンド撮像データ

スカイフラット、ダーク、バイアスの生データ

## ※詳細

積分時間、枚数	B	V	R	Ha	[OIII]	[SII]	フィルター無し	メモ
バイアス	バイアスなので積分時間は無し。2012/2/22に取得したもので代用しているが、バイアスは非常に安定しているので問題ないはず。(そもそもダークが撮れているのでデータ処理で使う必要性もない)							
ダーク							30s×5shots, 60s×5shots, 90s×5shots	
フラット (スカイ)	30s × 7shots	30s × 7shots	30s × 7shots	60s × 7shots	90s × 7shots	90s × 7shots		ファイル名の1stや2ndは撮った時間に応じて。Ha,[OIII],[SII]は入射光量的に質が悪い
M51	60s × 10shots	60s × 5shots	60s × 5shots	60s × 10shots	60s × 10shots	60s × 10shots		Bバンドはフィルター幅は広いが検出器の効率が悪いいため多めに撮っている。

# 使用手順

※サンプルデータのVバンド画像の一次処理

# 1、バイアスorダーク作成: mkdark.cl

- ロジック

複数枚のバイアスやダーク画像のメディアン画像を作る

- 作業 (dark/ディレクトリ内で)

まず複数の画像をリスト化 (積分時間が同じ画像だけを使うこと)

```
% cd dark/
```

```
% ls *dark30s*.fit > law-dark30s.lis
```

```
% ls *dark60s*.fit > law-dark60s.lis
```

作業ディレクトリにコピーした mkdark.cl を開いて以下のように編集

```
##### 書き換えるのはココだけ #####
```

```
fn1="law-dark30s.lis" # インプットリスト。dark 生データ #
```

```
fn2="dark30s.fits" # 出力画像名 #
```

```
#####
```

# 1、バイアスorダーク作成: mkdark.cl

Iraf 上で mkdark.cl を実行

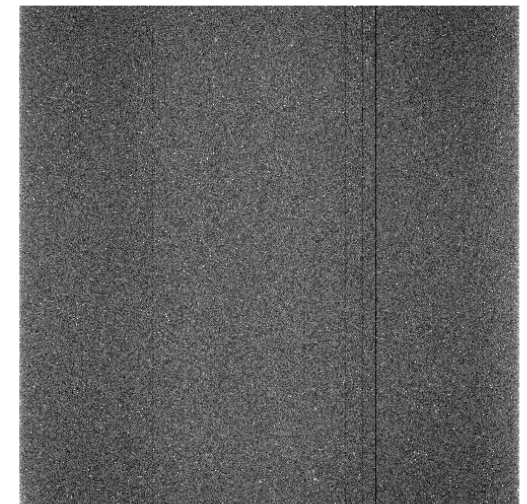
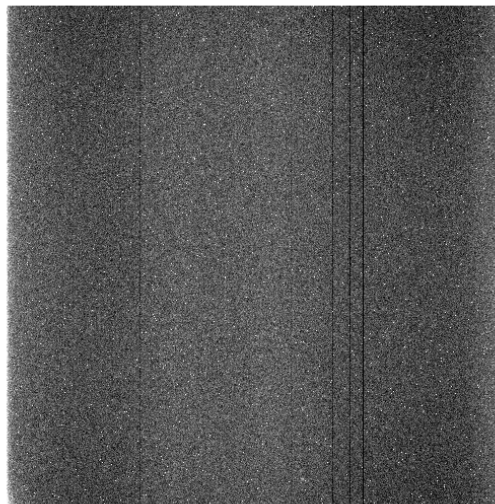
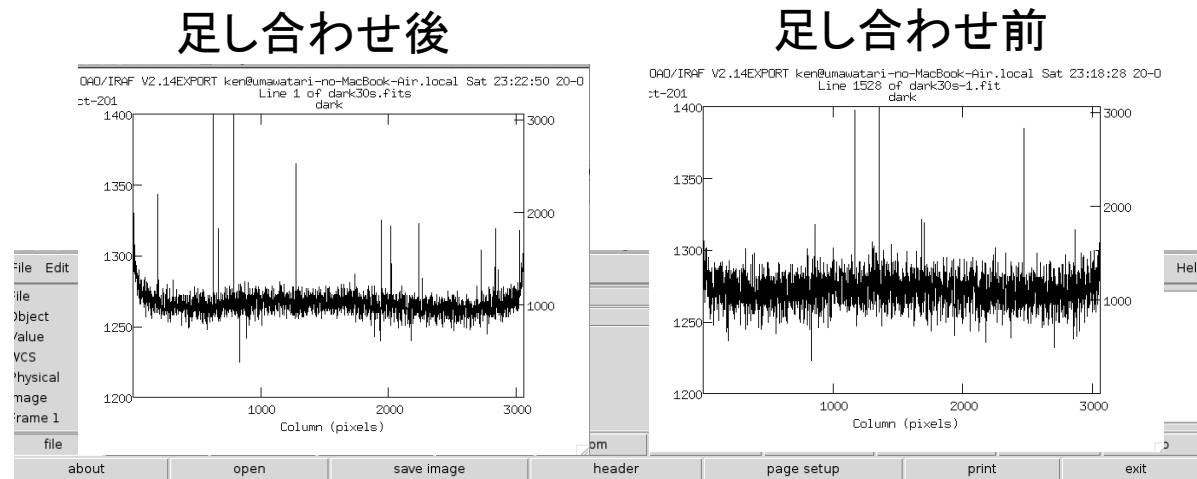
*ecl> cl< mkdark.cl*

同様に60秒分のダーク  
dark60s.fitsも作っとく。

- **結果**

足し合わせ前(右)と後(左)  
の画像、および $y=1500$ での  
断面図

断面図を見れば足し合  
わせによるノイズ減少の効果  
が分かる



## 2、フラット作成; mkflat.cl

- ロジック

フラット用画像複数毎に対して「ダーク画像引き => 規格化(画像のメディアン値を1にする) => 重ね合わせ」を一気に行う

- 作業(skyflat/ディレクトリ内で)

まずフィルター、積分時間毎に生フラット画像をリスト化(例えばVバンド、30s)

```
% ls skyflat*V30s*.fit > law-flatV30s.lis
```

作業ディレクトリにコピーした mkflat.cl を開いて以下のように編集

```
##### 書き換えるのはココだけ #####
```

```
fn1="law-flatV30s.lis" #インプットリスト。flat生データ#
```

```
fn2="../dark/dark30s.fits" #darkまたはbias画像# ←注意！積分時間が同じ物  
を使う事
```

```
fn4="skyflatV.fits" #出来上がりのflat画像名#
```

```
#####
```

## 2、フラット作成; mkflat.cl

Iraf 上で mkflat.cl を実行

*ecl> cl< mkflat.cl*

(※こけると色々いらないファイルがディレクトリ内に残ってしまい、2回目以降の使用の際に悪さするので、身に覚えのないファイルは消しといてください。)

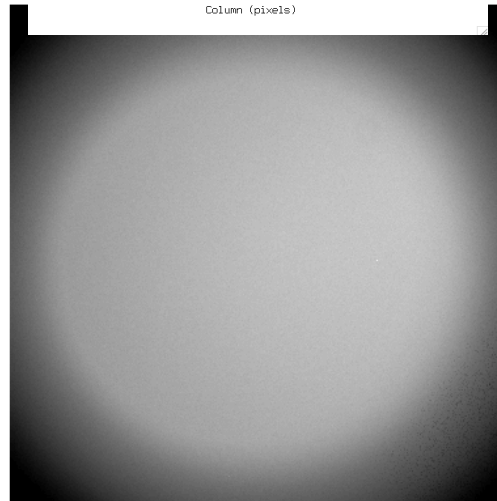
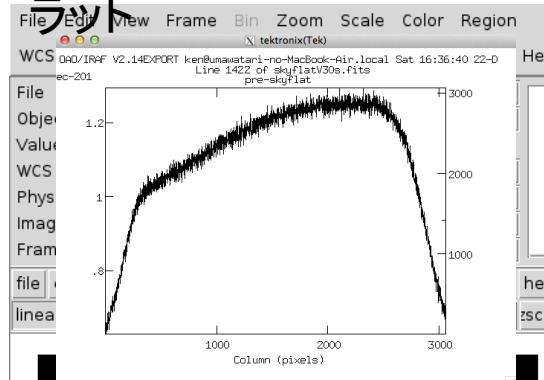
### • 結果

生画像(右)とフラット作成後(左)

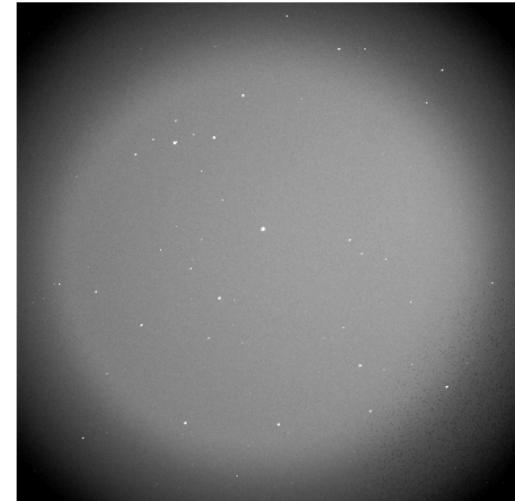
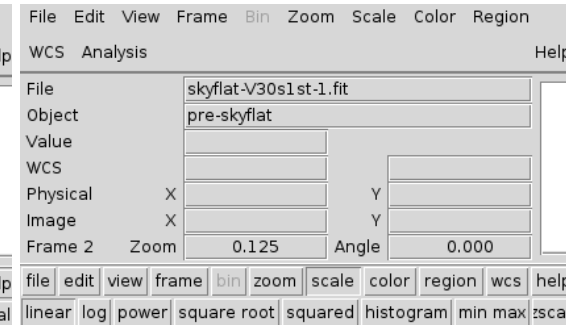
一枚一枚のスカイフラット生画像では星が見えるが多数毎の重ね合わせにより消えてるのが分かる。  
また規格化してあるので、完成フラット画像はカウントが1前後になっている。

フラットパターンは普通このように真中がよくはじが悪い。

ダーク引き・規格化・重ね合わせ後の完成スカイフラット



スカイフラット生画像



### 3、フラット割り: flatfield.cl

- ロジック

オブジェクトフレームから1で作ったダークを引き、2で作ったフラットで割る。複数毎分を一気に行う。(重ね合わせは今はしない)

- 作業(M51/ディレクトリで)

フィルター、積分時間毎に生オブジェクト画像をリスト化

```
% ls M51*V60s*.fit > lawM51V60s.lis
```

出来上がりの画像リストも先に準備(名前だけ、実体はなし)

```
% awk '{print "f"$1"s"}' lawM51V60s.lis > flM51V60s.lis
```

作業ディレクトリにコピーした flatfield.cl を開いて以下のように編集

```
##### 書き換えるのはココだけ #####
```

```
fn1="lawM51V60s.lis" #インプットリスト。生データ#
```

```
fn2="../dark/dark60s.fits" #darkまたはbias画像# <=積分時間に注意
```

```
fn4="../skyflat/skyflatV.fits" #skyflat#
```

```
fn5="flM51V60s.lis" #出来上がりの画像リスト名.あらかじめ作っておいてください#
```

```
#####
```



### 3、フラット割り: flatfield.cl

iraf上でflatfield.cl実行

```
ecl> cl< flatfield.cl
```

- **結果**

生画像(右)とフラット割り後(左)

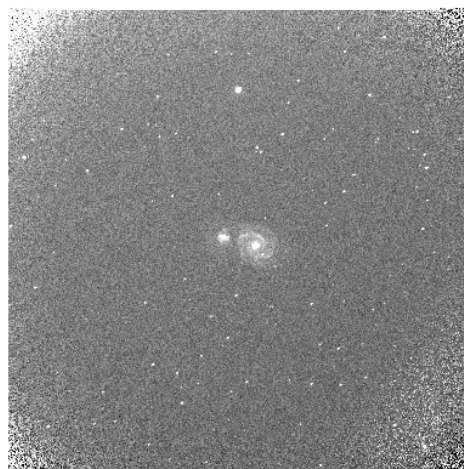
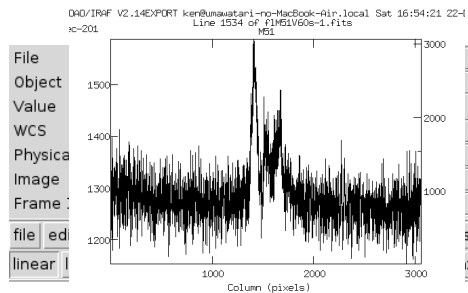
断面図を見ると生画像ではフラットパターンの上に銀河の光が乗っているのが分かる。

それがフラット割りによってバックグラウンドが平らになっている(端は元々感度悪いので仕方ない)。

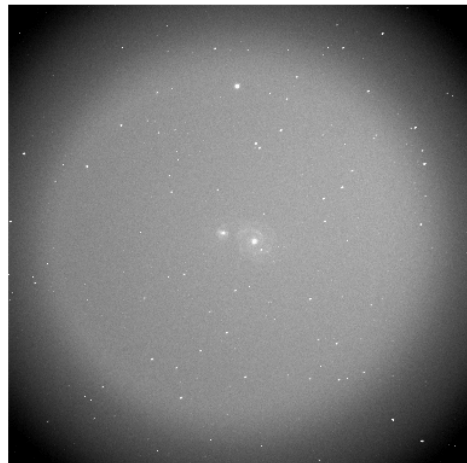
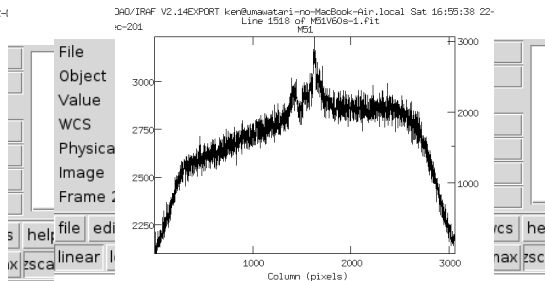
この例では少しだけ傾きが残ってしまったが、理想的には真っ平らになってほしい...

また注意してみればダークを引いた分、バックグラウンドのカウントが大体1000くらい少なくなっているはず。

dark引き・フラット割り後のM51画像(V)



M51生画像(V)





## 4、スカイ引き:subsky-simple.cl

### • ロジック

フラット割りまで終わったオブジェクトフレームに対して、スカイ引きを行い、天体が無い部分のカウントを大体0にする。画像内の任意の領域のメディアンカウントを取って定数引きするだけなので、3のフラット割りの精度が悪いと引け残り・引き過ぎが生じる。複数毎分を一気に行う。(重ね合わせは今はしない)

### • 作業(M51/ディレクトリで)

出力画像を先にリスト化(名前だけ。実体は無し。)

```
% awk '{print "subsky"substr($1,3,20)}' flM51V60s.lis > subskyM51V60s.lis
```

スカイカウント推定用画像リスト作成

```
% sed 's/fits/fits[401:1400,401:1200]/g' flM51V60s.lis > flM51V60s-sky.lis
```

※スカイカウントの推定にここでは各画像の $401 < x < 1400$ ,  $401 < y < 1200$ の領域を使った。メディアン値をスカイカウントとみなすので、よほど大きな天体が写ってない(もしくはフラット割りが上手くいってない)限りは画像全体を使っても同じだが、念のため。とにかくその領域のメディアンカウントが着目してる天体周りのスカイカウントになりそうな領域を指定すること。

## 4、スカイ引き:subsky-simple.cl

### • 作業

作業ディレクトリにコピーした subsky-simple.cl を開いて以下のように編集

#####書き換えるのはココだけ#####

fn1="flM51V60s.lis" #インプットリスト。flat-fieldingまでされた画像#

fn2="subskyM51V60s.lis" #出来上がりの画像リスト名.あらかじめ作っておいてください#

fn5="flM51V60s-sky.lis" #sky推定をfn1以外の画像で行う場合(例えば画像自体は同じだが使う領域を中心付近のみにするなど)、ここにそのリスト名を入れる。fn1と同じなら同じリスト名を入れる。#

#####

iraf上でsubsky-simple.cl実行

ecl> cl< subsky-simple.cl

## 4、スカイ引き:subsky-simple.cl

### • 結果

フラット割り画像(右)とスカイ引き後(左)

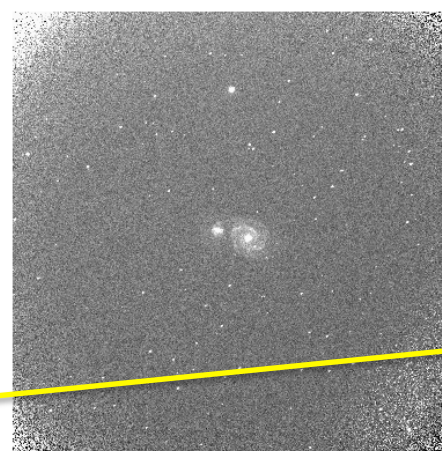
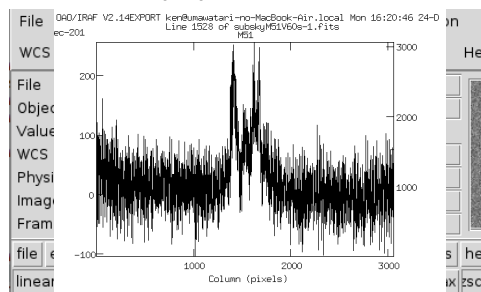
見た目には違いがないが、断面図を見ると、スカイ引き後はバックグラウンドのカウントが大体0になっているのが分かる。

あまりにも0から離れている場合には領域を変えるなどやり直した方がよいが、測光の時にもSExtractorでlocalなスカイ引きしてくれるので神経質になりすぎなくてもいい。

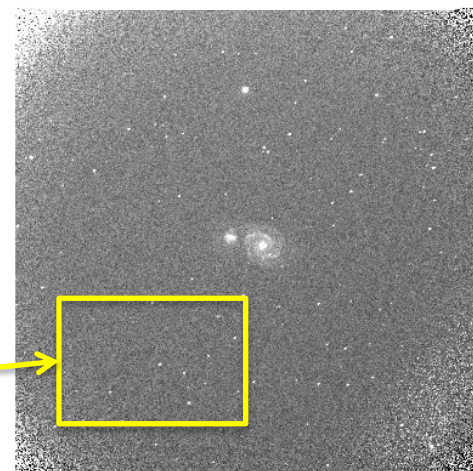
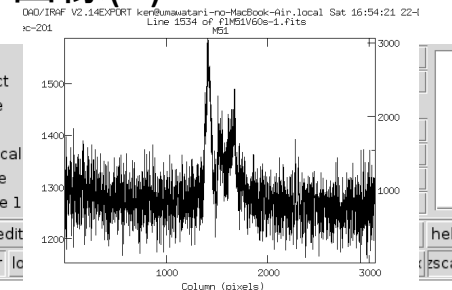
※おまけで各画像毎の時間とスカイカウントが格納されたskycount.datというファイルも出力される。

スカイ推定領域

スカイ引き後のM51画像(V)



フラット割り後のM51画像(V)



## 5、天体検出・カタログ作成: obj-detection.sh

- ロジック

この後の天体位置合わせの準備 => スカイ引きまで終わったオブジェクトフレーム一枚一枚に対して、SExtractorで天体検出をかけ、明るい点源カタログを作成。複数枚分を一気に行う。

- 作業 (M51/ディレクトリ内で)

SExtractorを使う関係上、適当でいいのでSExtractorを回すのに必要なパラメータファイル等3つのファイル(～.sex、～.conv、～.nmw)をコピーしておく。ここではSExtractorが用意してくれてる”default.\*”を使う。

```
% cp ~/sextractor-2.5.0/config/default.* .
```

※どのような基準で天体検出をかけるか(DETECT\_THRESH, DETECT\_MINAREA)はプログラムの中で別に指定するので、パラメータファイル(～.sex)は適当でいい。

## 5、天体検出・カタログ作成: obj-detection.sh

### • 作業

作業ディレクトリにコピーした obj-detection.sh を開いて以下のように編集

##### 書き換えるのはココだけ#####

#※適当でいいのでSExtractorのパラメータファイルとそれに応じた,~,conv,~,nmwを同じディレクトリ内に先に用意しておくこと。

SExt\_file="default.sex"

ID\_first=10

ID\_last=6

#天体検出かける画像を特徴づけるIDの始まりの文字数(ID\_first)とそこから終わりまでの文字数(ID\_last)# <= 今はsubskyM51V60s-1.fitsなどの画像に対して"V60s-1"をIDと考えれば、10文字目から6文字なので、ID\_first=10, ID\_last=6

#SExtractorのパラメータ設定# <= ケースバイケースで条件を加える事。

minarea=20 #DETECT\_MINAREA#

threshold=5 #DETECT\_THRESH# <= 「5 $\sigma$ 以上のカウントが20ピクセル以上連なったら天体」という設定

aperture=20 #pixel単位で測光aperture指定#

(次ページに続く)

## 5、天体検出・カタログ作成: obj-detection.sh

#出力parameterリストobj-detection.paramの作成。出力させたいparameterを自由に設定# <=アルゴリズム的には「x座標、y座標、明るさ、FLAGS」だけあれば十分なのでここはこのままでOK。改良の余地として、星かどうかを判断する「CLASS\_STAR」を入れれば良かったかも・・・

```
if [ -e obj-detection.param ]; then
    rm obj-detection.param
fi
echo "NUMBER" > obj-detection.param
echo "X_IMAGE" >> obj-detection.param
echo "Y_IMAGE" >> obj-detection.param
echo "FLUX_APER(1)" >> obj-detection.param
echo "FLUX_AUTO" >> obj-detection.param
echo "BACKGROUND" >> obj-detection.param
echo "FLAGS" >> obj-detection.param
```

(次ページに続く)

## 5、天体検出・カタログ作成: obj-detection.sh

#天体カタログの位置しぼり、明るさしぼりの設定(必要なければ全部defaultにしとくこと)# <=>とは言っても実際の画像ではS/Nの悪い端の方で偽天体を検出し易かったり、明るすぎる天体はサチっていたりするので、領域をしぼったり一番明るい天体は外したりする工夫を施した方がいいと思う。

```
x_min=350  #(0)#
```

```
x_max=2700  #(3056)#
```

```
y_min=350  #(0)#
```

```
y_max=2700  #(3056)#
```

```
luminous1=5  #明るい方の"luminous1"番目から(1)#
```

```
luminous2=30  #"luminous2"番目までを選んでくる(10000)#
```

```
###書き換えるのココまで#####
```

ターミナル上でobj-detection.shを実行

```
% chmod u+x obj-detection.sh
```

```
% ./obj-detection.sh subskyM51V60s.lis
```

※かならず天体検出をかけたい画像リストを"./obj-detection.sh"の後に続ける事

## 5、天体検出・カタログ作成: obj-detection.sh

### • 結果

作業ディレクトリ内に"detect-V60s-\*.cat"(\*の部分は数字)という天体カタログが出来てるはず。中身は

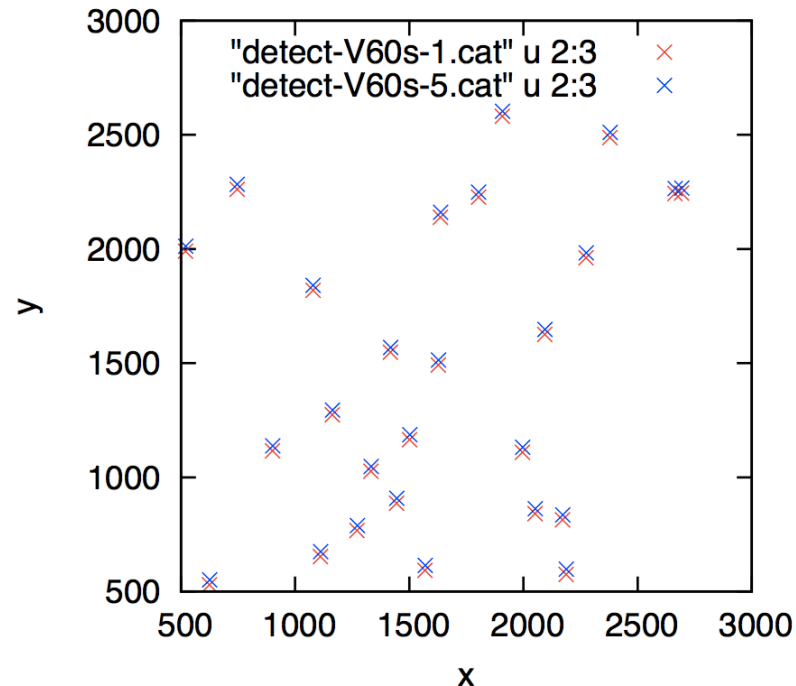
```
: 328 1802.977 2227.117 464857.5 473464.8 16.28401 0
   180 519.817 1990.109 400175.3 404612.8 22.41453 0
   119 1568.409 592.623 395295 399325.4 -8.457658 0
```

(以下略)

のように「番号、x、y、開口測光値(カウント)、クローン測光値、バックグラウンド値、フラグ」の情報が入っている。

gnuplotなどでこれらのファイルのx,yをプロットしてみて、異なる画像間で同一天体を拾っているようだったら成功。

また銀河のように広がった天体を拾ってしまっていないかもチェックポイント。(30個中の1個か2個くらいなら問題ない。)





## 6、画像の位置合わせ : position-match.cl

### • ロジック

スカイ引きまでした画像は、画像毎に天体の写っているxy座標が異なるため、そのまま足し合わせてもブレた画像になってしまう。そこで5で作成した天体位置情報カタログから、全画像をある特定の一枚に天体座標が合うように変換する(位置合わせ)。具体的には

(1)天体カタログから同一天体マッチング(xyxy-match)

(1')画像のフォーマット広げる(平行移動した分が切り取られないように)

(2)画像の変換規則作り(geomap)

(3)実際に画像変換(geotran)

の3段階からなる。

### • 作業(M51/ディレクトリ内で)

入力天体カタログを先にリスト化(スカイ引き後の画像リストとは別)

```
% ls detect-V60s-*.cat > detect-V60s-cat.lis
```

出力画像リストを先に準備(名前だけ。実体無し)

```
% awk '{print "shift"substr($1,7,20)}' subskyM51V60s.lis > shiftM51V60s.lis
```

## 6、画像の位置合わせ : position-match.cl

### • 作業

作業ディレクトリにコピーした position-match.cl を開いて以下のように編集

```
####書き換えるのはココだけ#####  
#ただしxyxymatchはこけやすいのでそのパラメータは適時いじるべし###  
inputimage_lis="subskyM51V60s.lis" #位置合わせ前の画像リスト#  
inputimagefits="subskyM51V60s-1.fits" #位置合わせ前の画像を適当に1枚。フォーマット  
調べる用#  
inputcat_lis="detect-V60s-cat.lis" #天体カタログが入ったリスト#  
xcol=2 #天体カタログの何列目にxが書かれているか#  
ycol=3 #天体カタログの何列目にyが書かれているか#  
refcat="detect-V60s-3.cat" #xyxymatchのリファレンスとなるカタログ名#  
deltax=100 #フォーマット拡張でx方向に増やしたいピクセル数#  
deltay=100 #フォーマット拡張でy方向に増やしたいピクセル数#  
min_match=10 #天体マッチ数の最小値。これより少ない天体数しかマッチしな  
かったら(1)で中止する#  
outputimage_lis="shiftM51V60s.lis" #出来上がりの画像リスト名.事前に用意しとく。#  
#####書き換えここまで#####
```

## 6、画像の位置合わせ : position-match.cl

iraf上でposition-match.cl実行

ecl> cl< position-match.cl

全タスクの中で一番失敗し易い所  
なので5も含めて慎重に行うべし

※とにかく(1)xyxymatchがコケ易いため、xyxymatchで同一天体マッチングが設定値以下の場合には“xyxymatch is failed!(Matched number is too small.) Program is stopped before geomap.”のエラー文とともにプログラムを中断するようにしている。

(確認方法)=>ds9に出来上がった5枚を表示させて“Tab”キー連打すると1フレームずつ画像が切り替わる。上手く位置合わせされていると天体が全く動かないように見えて気持ちいい。

※上手く位置合わせされていない場合のチェックポイント

- ①matchedOBJnumber.datを見て天体マッチ数を確認
  - ②5の作業で作ったカタログの26天体はファイル毎に大体同じかどうか
  - ③5の作業の天体数を増やしてやり直し
  - ④画像の変換規則の書かれたgeomap\_result.dbの中身を確認
- } xyxymatchでコケた場合

## 7、重ね合わせ: final\_imcomb.cl

### • ロジック

6まで天体光だけが乗った画像かつ天体のxy座標が合わされたものが出来ているので、それらを一枚の画像に重ね合わせる。屋上で観測する時はフレーム数が少なかったりするので、デフォルトの重ね合わせ方はmedian。10枚以上撮っているような場合はmean又はclipped meanに変えても良い。

### • 作業 (M51/ディレクトリ内で)

作業ディレクトリにコピーした final\_imcomb.cl を開いて以下のように編集

#####書き換えるのはココだけ#####

fn1="shiftM51V60s.lis" #インプットリスト。dark生データ#

fn2="finM51V60s.fits" #出力画像名#

fn4="exp\_M51V60s" #exposuremapの名前。拡張子はいらない#

#####

※重ね合わせ方を変えたければ、プログラム内imcombのパラメータの  
combine="median"の部分を適時変えて下さい。

# 7、重ね合わせ : final\_imcomb.cl

iraf上でfinal\_imcomb.cl実行

```
ecl> cl< final_imcomb.cl
```

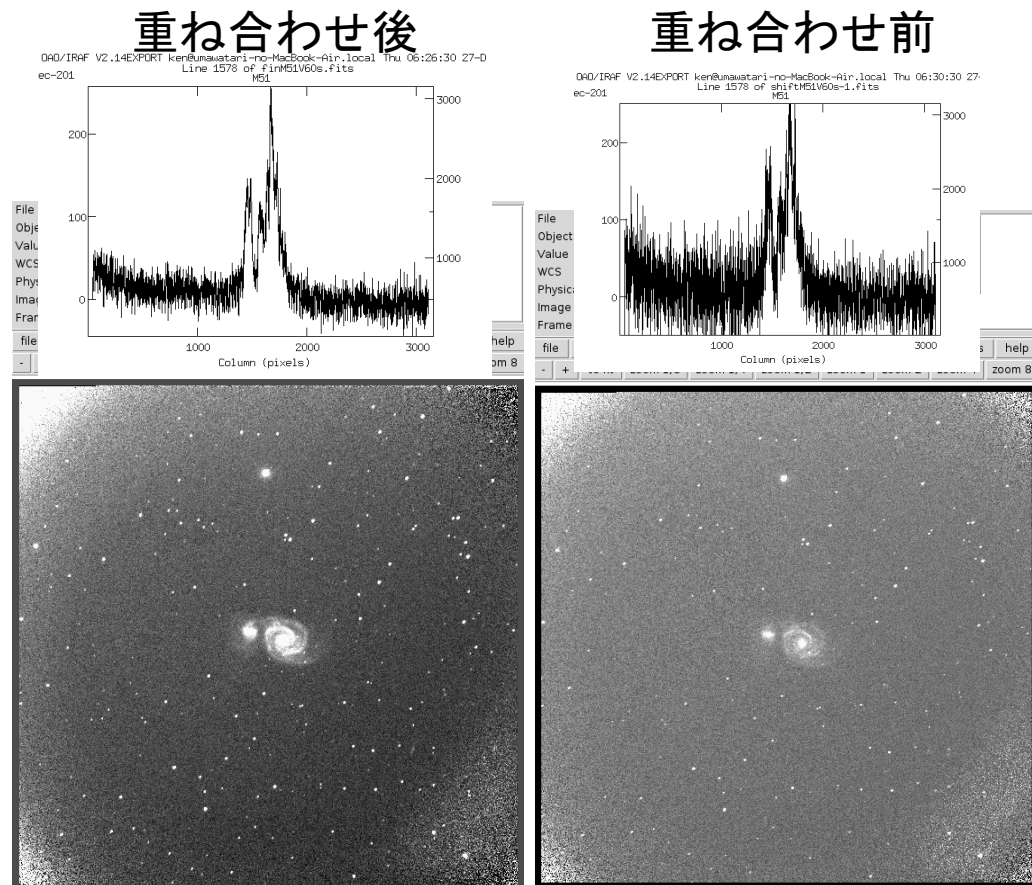
- 結果

重ね合わせ後(右)と前(左)

重ね合わせる前にかすかにしか写っていないような暗い天体でも、重ね合わせ後の画像ではしっかり写っていたりするはず。

断面図を見ると明らかにS/Nが良くなった事が分かる。

(※そのぶん、フラット割りが完璧でなく若干バックグラウンドが傾きを持ってしまったことも強調されたが・・・)





# 小休止：ゼロ点決め、WCS貼付け

プログラム化していないが、重要な2つの作業を簡単に紹介

## ・ゼロ点決め

カウントと等級の関係規則を標準星を使って作る。

サンプルデータには専用の標準星画像はないが、  
右図の3つの星のV等級はSIMBADより

13.3(～J13303～)、13.4(～J13301～)、12.4(～J1329～)

またfinM51V60s.fitsでこれらの星の測光を行うと

65920 (Count, 20pix aperture, ～J13303～)、

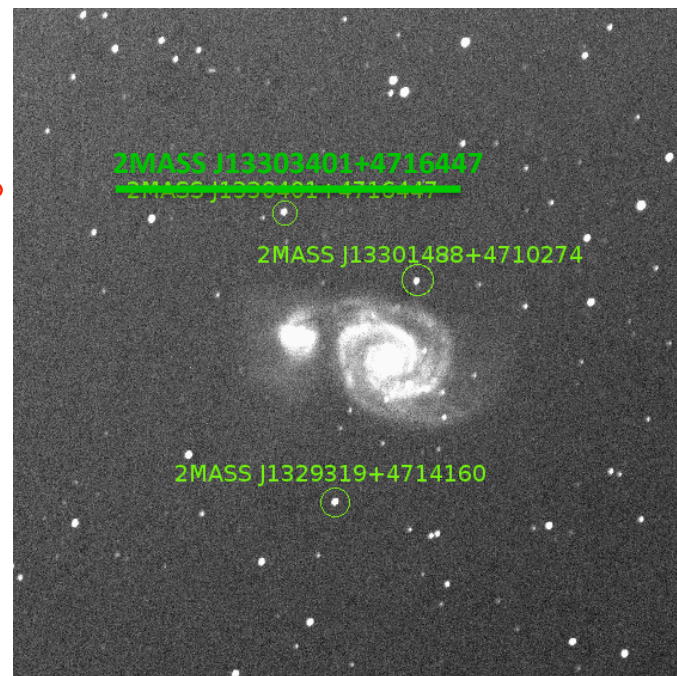
54518 (～J13301～)、102082 (～J1329～)

これをゼロ点の定義式

$m_{\text{std}} = -2.5 \log(\text{COUNT}_{\text{std}}) + \text{ZERO}$  に代入すればゼロ点が求まり、  
3つの星から求めた平均のゼロ点は

**25.2**

となる。(ただし20pix apertureの場合の60秒分のゼロ点である事に注意)



# 小休止：ゼロ点決め、WCS貼付け

プログラム化していないが、重要な2つの作業を簡単に紹介

## ・WCS貼付け

World Cordinate System、簡単には「**赤経・赤緯**」を画像に貼付けること。

画像内にある赤経・赤緯の分かっている星を使って、

**(x,y)装置座標 => (R.A.,Dec.)座標 の変換関係を求めて適用**すればいい。

USNOカタログなどがよく使用される。

IRAFで作業するなら、

複数天体の「x,y,ra,dec」の対応リストを作りccmapとccsetwcsというタスクを使う。

基本的には以上で「一次処理」は終わり

## 8、仮想スカイ画像作成: mknoiseimage.cl

### • ロジック

この後の「画像の限界等級決め」の際に使う、バックグラウンドノイズだけで埋めつくされた「仮想スカイ」画像を作る。一時処理済み画像の天体の部分に画像の他の部分をあてはめる。それを複数毎作って重ね合わせることで画像から天体を消し去る。

### • 作業

まず最初に画像の天体部分が+、それ以外の部分が0になっているようなマスク画像を手作業で作る。

例)

```
% cp default.sex mk_mask.sex
```

なるべく天体を多めに&大きめに拾うようにパラメータ調整。

```
⇒ DETECT_MINAREA 5、DETECT_THRESH 2、CHECKIMAGE_TYPE  
SEGMENTATION、CHECKIMAGE_NAME mask_finM51V60s.fits
```

など。(CHECKING\_TYPEは必ずSEGMENTATIONにすること！)

マスク画像(SEGMENTATIONファイル)作りたいだけなので、測光に関するパラメータなどには注意しなくてOK。



## 8、仮想スカイ画像作成: mknoiseimage.cl

### • 作業

SExtractor実行

```
% sex finM51V60s.fits -c mk_mask.sex
```

mask\_finM51V60s.fitsというファイルができていますので、finM51V60s.fitsと並べてds9でチェック。

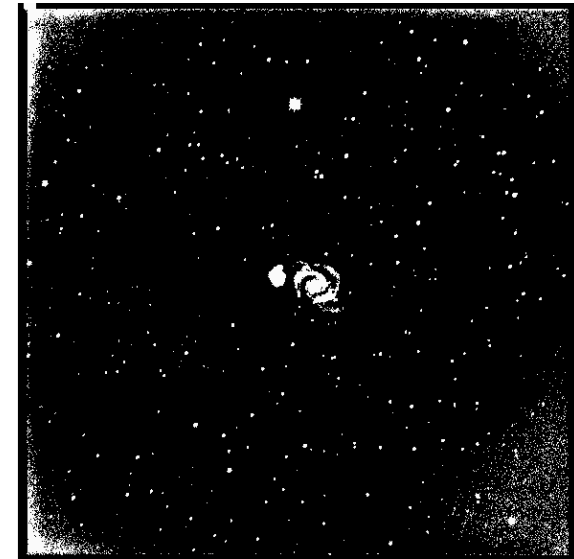
天体の部分白く(+)になっていたらOK

ノイズを拾いすぎていたら  
やり直し(はじのS/Nの悪い  
所はどうしようもない・・・)

(finM51V60s.fits)



(mask\_finM51V60s.fits)



マスク作りは非常に重要かつケースバイケースなのでスクリプト化せず、手作業にした。紹介したのは一番単純なやり方。

後でもう少し工夫したやり方も紹介する。

## 8、仮想スカイ画像作成: mknoiseimage.cl

### • 作業

マスク画像作ったら、作業ディレクトリにコピーした mknoiseimage.cl を開いて以下のように編集

#####書き換えるのはココだけ#####

inimage="finM51V60s.fits" #画像の元となる画像#

outimage="noise\_finM51V60s.fits" #出力するノイズ画像名#

maskimage="mask\_finM51V60s.fits" #maskイメージ。あらかじめ作っておく#

nit=10 #ループ数。画像内に天体がたくさん写っている場合大きくした方がいいかも#

#####

iraf上でstdsdasパッケージに入っておく

iraf上でmknoiseimage.cl実行

stdsdas> cl< mknoiseimage.cl

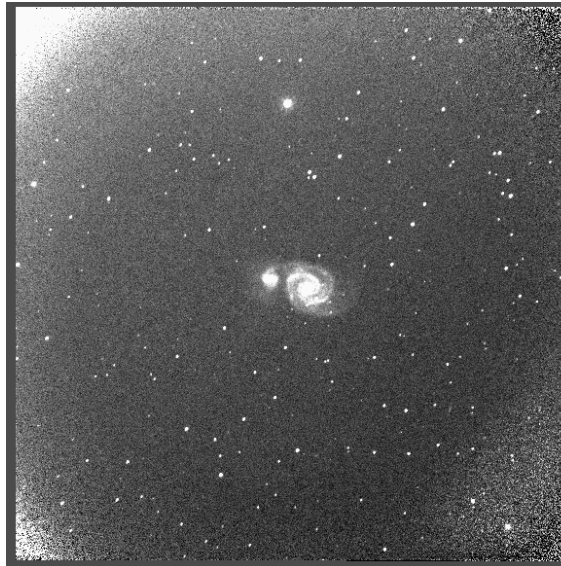
## 8、仮想スカイ画像作成: mknoiseimage.cl

### • 結果

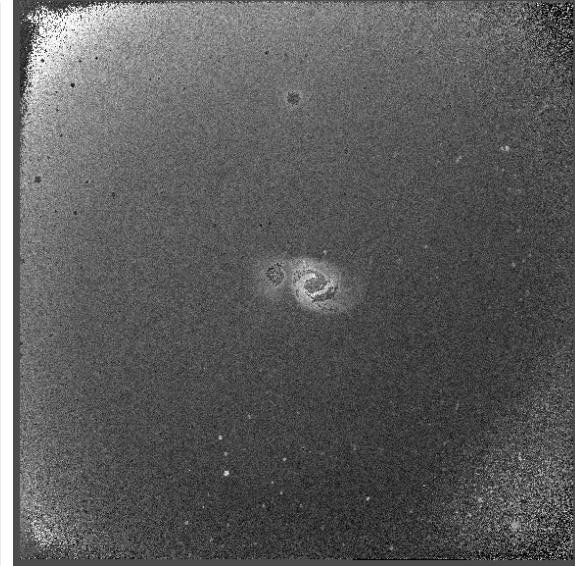
一次処理済の天体画像(左)と  
仮想スカイ画像

基本的には星や銀河中心など  
カウントが非常に大きい部分は  
スカイで埋められているので、  
どこを測光しようとスカイの  
カウント(0程度)を返すような画  
像になっている。

天体画像



仮想スカイ=  
天体部分をスカイで埋めた  
ノイズ画像



ただ以下の2点の問題がある

(1) 銀河の淡く広がった成分やS/Nの悪い端はカバーできてない。=> マスクをもっと工夫(次スライド)

(2) 元々の画像でスカイに傾きが  
あったため、周辺に比べて明るく/暗く埋められていたりする。=> スカイの傾きがリアルなものであればしょうがない。が、この後の検出限界決定には大きく影響しないはず。

## 8、仮想スカイ画像作成(+ $\alpha$ 、マスクの工夫)

マスクとしてSExtractorのSEGMENTATIONファイルを使うだけだと、淡い成分や画像のへりに対して不十分。

### (改良例)

irafのmkpatt(artdataノパッケージ)というタスクでfinM51V60s.fitsと同じフォーマット(3156×3156)でカウントは1の画像を作成

```
stsdas> artdata
```

```
artdata> mkpatt NEOmask_finM51V60s.fits pattern=constant v1=1 size=1  
pixtype=real ndim=2 ncols=3156 nlines=3156
```

前のmask\_finM51V60s.fitsからマスクとして使っていい部分(中心のM51とへり以外)をNEOmask\_finM51V60s.fitsにコピー

```
artdata> bye
```

```
stsdas> imcopy mask_finM51V60s.fits[301:1300,101:2800]  
NEOmask_finM51V60s.fits [301:1300,101:2800]
```

```
stsdas> imcopy mask_finM51V60s.fits[2001:2900,201:2900]  
NEOmask_finM51V60s.fits [2001:2900,201:2900]
```

## 8、仮想スカイ画像作成(+ $\alpha$ 、マスクの工夫)

(改良例) 続き

```
stdas> imcopy mask_finM51V60s.fits[1301:2000,1801:3110]
```

```
NEOmask_finM51V60s.fits[1301:2000,1801:3110]
```

```
stdas> imcopy mask_finM51V60s.fits[1301:2000,51:1300]
```

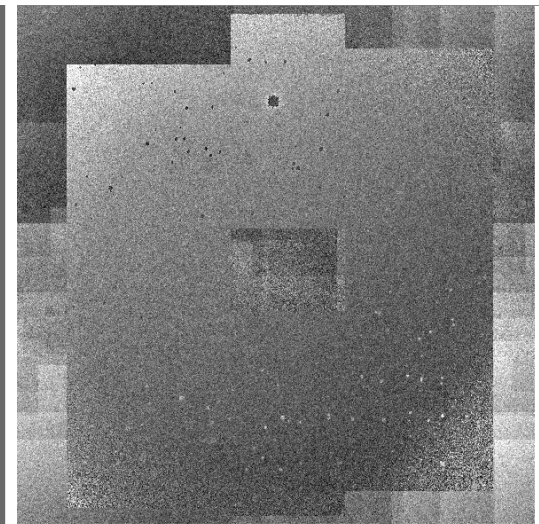
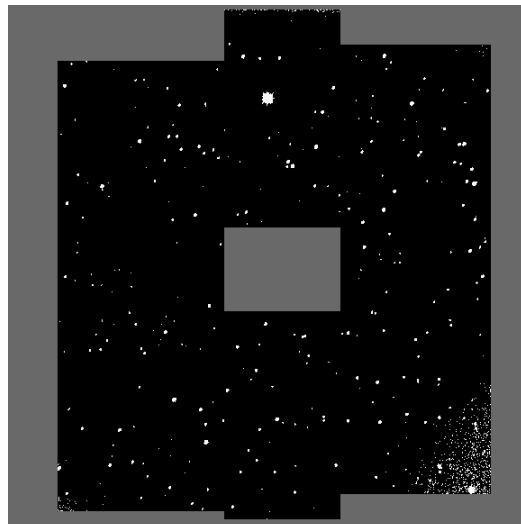
```
NEOmask_finM51V60s.fits[1301:2000,51:1300]
```

できた新しいマスク画像が下左図。

また新しいマスク画像を使って作った仮想スカイ画像が下右図。

人工的な感じに見えるが仕方ない。

そもそも元々の画像でS/Nの悪い  
へりを切り取って、それ以後切り  
取った画像しか使わない事にして  
しまえば、より自然(かつ現実的)  
かも。。





## 9、画像のノイズレベル決め: estbackerr.cl

- ロジック

バックグラウンドノイズのゆらぎの平均と分散を求める。具体的には「仮想スカイ画像のランダムな場所での天体測光」を多数回行い、平均する。ここで得られたバックグラウンドの分散( $1\sigma$ )とゼロ点から限界等級も推定できる。

- 作業

作業ディレクトリにコピーした estbackerr.cl を開いて以下のように編集

#####書き換えるのはココだけ#####

imobj = "finM51V60s.fits" #天体画像#

imnoise = " NEOnoise\_finM51V60s.fits" #仮想sky画像#

fsex = "detect.sex" #出力ファイルやparamファイル,Apertureなど最低限のパラメータはあとで指定される値で上書きされるが、それ以外のパラメータも適切なものにしたい方がいい#

(次ページに続く)

## 9、画像のノイズレベル決め: estbackerr.cl

```
output = "bkgnoise_finM51V60s.txt" #最終的に欲しいskyの平均カウントとそこ  
からの標準偏差 $\sigma_{\text{sky}}$ の入るファイル名#  
check_head = "checkbkg_finM51V60s" # ***.txt => 全ランダムアパーチャにおけ  
る測光値ファイル, ***.hist => 全測光値を明るさ別にヒストグラムにしたファイル  
***.eps => チェック用の図  
sigma=5 #何 $\sigma$ 限界を求めたいか?#  
zero=25.2 #画像のゼロ点(1秒あたりでなく積分時間あたり)#  
  
#SExtractorのパラメータ設定#  
aperture=20 #apertureの設定#  
pscale=1.38 #1pixelを見込む角度(arcsec)# <=サンプルデータの場合  
minarea=10 #DETECT_MINAREA#  
threshold=5 #DETECT_THRESHOLD# <=一応まともな天体検出をしそうな値を設定  
しておくこと  
nit = 10000 #合計何点skyのデータをとるか?# <=多いほどいいが時間もかかる  
#####書き換えるのここまで#####
```

# 9、画像のノイズレベル決め: estbackerr.cl

iraf上でestbackerr.cl実行

```
stdas> cl< estbackerr.cl
```

cycle No.1 total number = 626

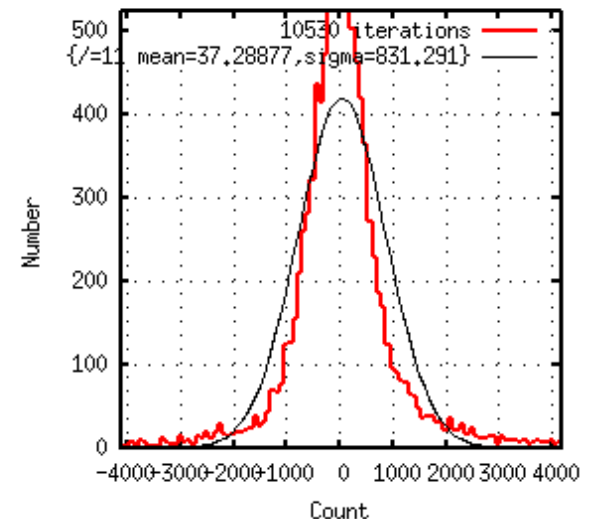
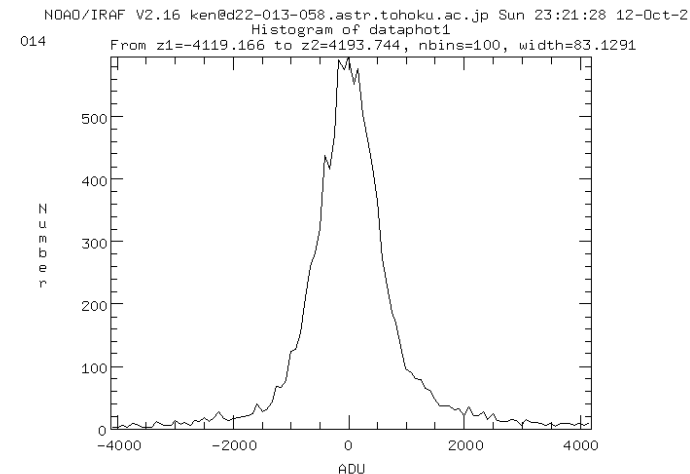
.....

aperture測光を多数回やっていく過程が表示され、total numberが10000回を超えたら、その10000回分の測光値の平均と標準偏差を求める。その時に、観測された頻度ヒストグラム(iraf画面)とそれに対するガウス関数(gnuplot)も表示される。

観測頻度ヒストグラムがガウス関数的になっていればOK  
iraf端末内で"Enter"を押すとプログラム終了  
測光値、ヒストグラムのファイルと図が保存される

右の場合、実際の分布がガウス関数よりも裾野を引いた感じで上手くない(背景ゆらぎを過大評価してる)。

(1)画像そのもの(2)SExtractorのパラメータ(3)プログラム内ガウス関数フィッティングの部分(imstatのパラメータ)などについて更なる考察が必要。。。





## 9、画像のノイズレベル決め: estbackerr.cl

### • 結果

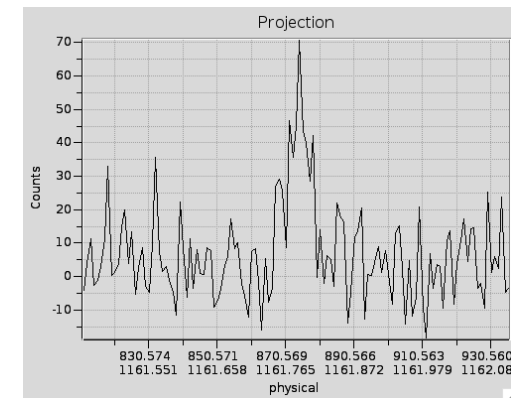
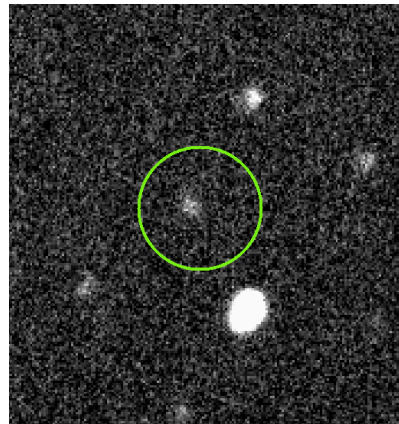
結果の入ったbkgnoise\_finM51V60s.txtの中身を見してみる

```
% less bkgnoise_finM51V60s.txt
```

```
# Name aperture(pix) aperture(") stddev(ADU) mean(ADU) 5 sigma limit  
finM51V60s.fits 20.000 27.600 831.2910 37.28877 16.153
```

これは20pixel aperture (27.6'')で測光する場合のバックグラウンドノイズのゆらぎ $1\sigma$ が831 countで、そこから $5\sigma$ 限界等級を計算すると16.2(mag)という事。

ちなみに画像内で16.2mag程度のカウントを持つ天体は下図のようなもので確かに $5\sigma$ 程度で受かっている。



# 10、画像のPSF決め: est\_seeing.cl

- ロジック

画像にSExtractorをかけて星らしいものを複数選び出す。それらのPSF(FWHM)を測って、mean,midpt,modesなどをその画像のPSFとして出力する。

- 作業

まずPSF測りたい画像をリスト化(1枚しかなくてもリスト化する)

```
% ls finM51*.fits > finM51.lis
```

作業ディレクトリにコピーした est\_seeing.cl を開いて以下のように編集

```
#####書き換えるのはココだけ#####
```

```
objlis = "finM51.lis" #天体画像リスト、用意しとく#
```

```
output = "finM51_PSF.data" #最終的に欲しいskyの平均カウントとそこからの  
標準偏差 $\sigma_{sky}$ などが入るファイル名#
```

(次ページに続く)

# 10、画像のPSF決め: est\_seeing.cl

#(1)SExtractorのパラメータ設定#

conf\_sex = "detect.sex" #出力ファイルやparamファイル,Apertureがどうであ  
れあとで全部指定されるのでココに入るパラメータファイルはなんでもいい#

aperture=10 #apertureの設定#

pscale=1.38 #1pixelを見込む角度(arcsec)#

minarea=10 #DETECT\_MINAREA#

threshold=5 #DETECT\_THRESHOLD#

zero=25.2 #画像のゼロ点(1秒あたりでなく積分時間あたり) 今は適当#

seeing\_sex=10 #[arcsec]最初にFWHMの見当値を入力しとく(大体でいい)

strra\_min=100 #各画像から星を選び出す時の位置しぼり. X座標.Image座標#

strra\_max=2900

strdec\_min=100 #y座標#

strdec\_max=2900

(次ページに続く)

# 10、画像のPSF決め: est\_seeing.cl

```
#(2)星しばりの設定:屋上望遠鏡の場合はゆるめにした方が良さげ#  
strflag_max=1 #星として使う天体のフラグ許容範囲:この値より小さい(!<!)#  
strclass_min=0.8 #sextractorでのclass starの許容最小値#  
strellip_max=0.2 #ellipticityの許容最大値#  
strnum_max=100 #使う星の数の上限#  
#####書き換えるのここまで#####
```

iraf上でestbackerr.cl実行

```
cl> cl< est_seeing.cl
```

↓

finM51\_PSF.data・・・入力画像毎のPSFデータ

str\_?.cat・・・入力画像内の星データ

allobj\_?.cat・・・入力画像内の全星データ

:「No、aperture等級、kron等級、kron半径、bkg、x、y、ellipticity、FWHM(pix)、  
flag、class\_star」

# 10、画像のPSF決め: est\_seeing.cl

- 結果

結果の入ったfinM51\_PSF.dataの中身を見してみる

```
% less finM51_PSF.data
```

```
# No name used_str_num mean midpt mode stddev min max
1 finM51B60s.fits 97 11.4035 11.1399 10.7874 1.36438 9.21840 18.5334
2 finM51R60s.fits 100 11.5248 11.3667 10.8761 1.03647 9.67380 15.3318
3 finM51V60s.fits 47 12.8739 12.5916 12.5133 1.64395 10.1844 20.4654
```

それぞれの画像のPSFは

B => 11.4(",mean) 11.1(",midpt) 10.8(",mode)

V => 12.9(",mean) 12.6(",midpt) 12.5(",mode)

R => 11.5(",mean) 11.4(",midpt) 10.9(",mode)

これらの値の信頼性に関しては、次ページに示すようなチェックが必要。

# 10、画像のPSF決め: est\_seeing.cl

## • 結果

結果の妥当性のチェック

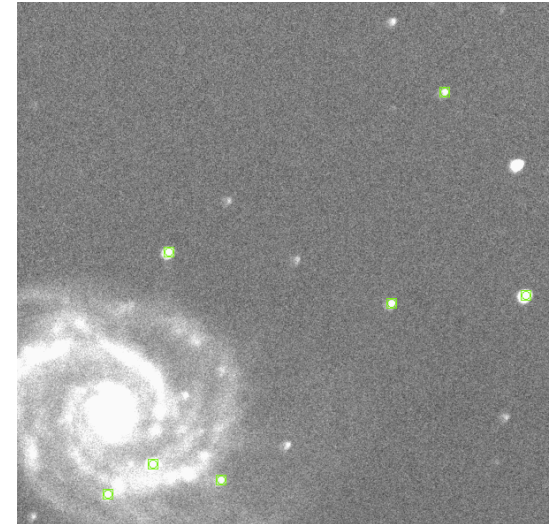
(1) 星と思っている天体を画像上でチェック

str\_?.catから(x,y)情報を抜き出して、ds9でチェック =>

(2) 全天体と星のFWHM分布を比較

str\_?.catとallobj\_?.catの等級とFWHM(")を

図示

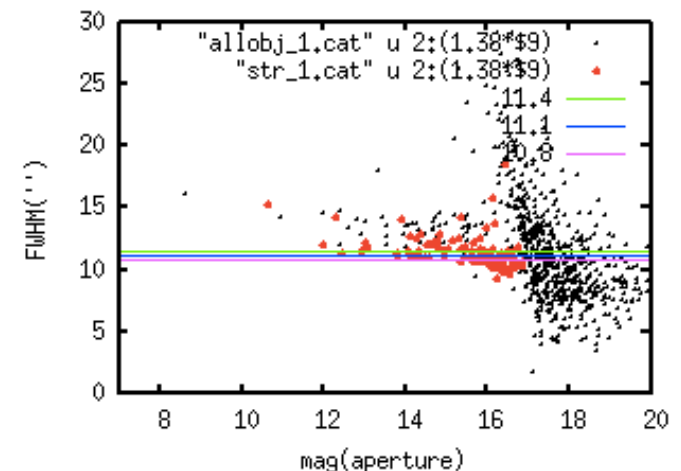


```
gnuplot> plot [7:20][0:50]"allobj_1.cat" u 2:(1.38*$9) pt 7 ps 1 lc 0 , "str_1.cat"
u 2:(1.38*$9) pt 7 ps 1 lc 1, 11.4 lc 2, 11.1 lc 3, 10.8 lc 4
```

右図の例だと、星選びは上手くできてるっぽい。

mean/midpt/modeはどれでもいいっぽい。

※屋上データだと収差のせいで星が綺麗な円形になっておらず、PSF求めが意味あるか不明・・・



# おまけ

## 3色合成



# 3色合成

三色合成は科学的に意味のある作業ではないが、ds9で簡単にできる。

(1) 3色分の一次処理済み画像を作る

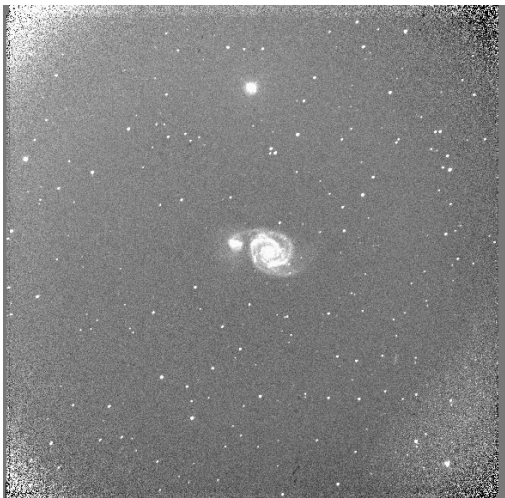
: Vはもう作ったので、例えばあとB,Rを揃える。

(2) 3色分の画像を位置合わせしておく

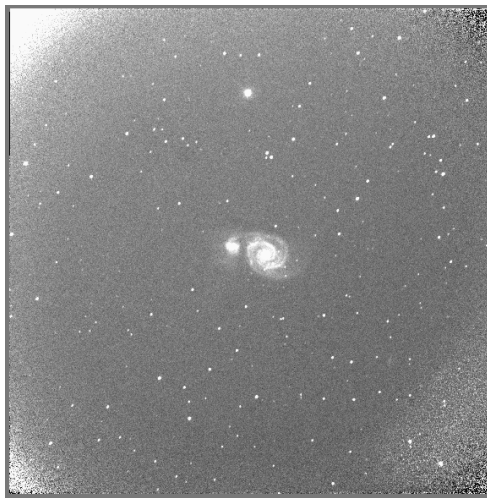
obj-detection.shとposition-match.cl使う。

=> finM51B60s+.fits、finM51V60s+.fits、finM51R60s+.fits

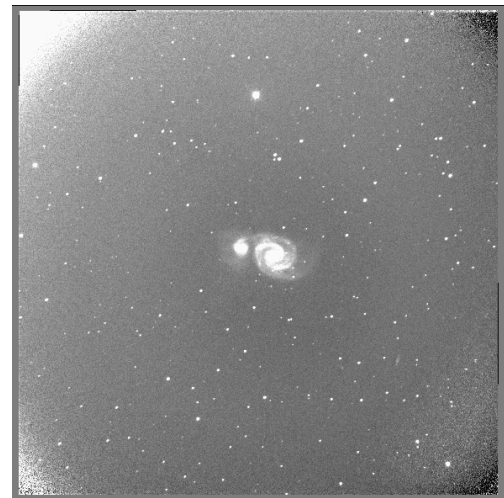
B



V



R



# 3色合成

(3) ds9で合成

“Frame”タブ=>“New Frame RGB”から3色合成用フレームを開き、

RGB小ウィンドウで選択された色 (red/Green/Blue) に対応した画像を“File”タブ=>“Open”で開いていく。

3色とも開いたら、各色毎に

“Scale”タブ=>“Scale Parameters...”でスケール調整して綺麗にしていく。

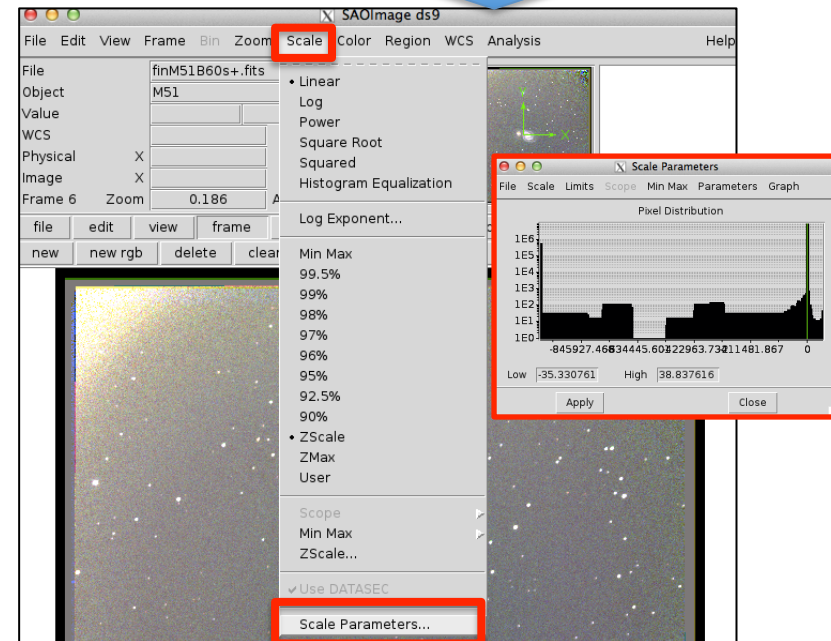
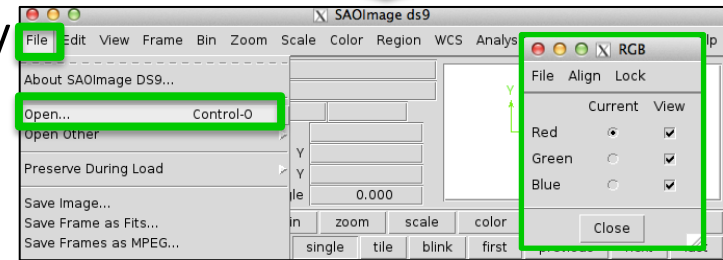
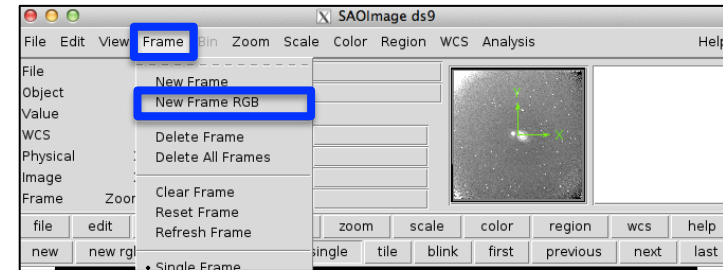
サンプルデータのM51の場合、

B: Low=0, High=220

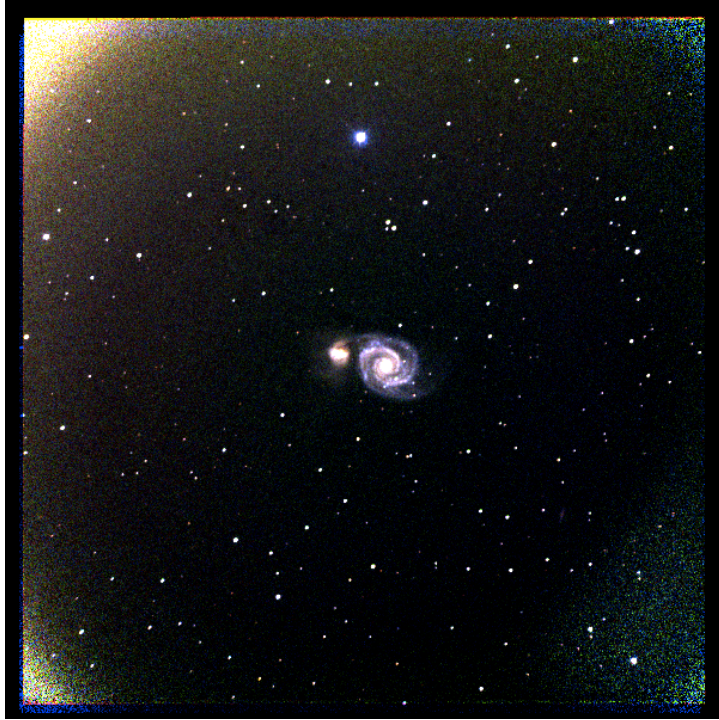
V: Low=0, High=180

R: Low=0, High=45

くらいでそれっぽく見える。



# 3色合成



M51を  
ズーム =>



おわり