Broadband Image Effective Frequency

W. D. Cotton (NRAO), November 20, 2024

***Abstract*—Issues involved with obtaining consistent and well defined effective frequencies in broadband continuum images derived from a set of subband images is discussed. A solution implemented in the Obit environment is presented.**

***Index Terms*—Image Effective Frequency**

I. Introduction

IDEALLY the broadband continuum image obtained from a set of narrower band images would be derived from a spectral fit in each pixel which possibly includes a spectral index and will have a well defined, and constant, effective frequency for all pixels. Unfortunately, in most images most pixels lack adequate sensitivity for this and an average of the channels is needed. Since the sensitivity of the various narrowband planes is generally not the same, and the spectral index of the sources in the field vary, some weighted average is needed. The details of the weighting determines the effective frequency of the broadband image and, if uncontrolled, can lead to a poorly defined effective frequency. This memo describes a technique for forming a wideband continuum image, possibly with spectral terms with a constant and well defined effective frequency using the Obit package [1][1] .

II. Effective Frequency

For wideband data which are imaged as multiple frequency subbands, the effective frequency of the resultant wideband image depends on how the individual frequency subbands are combined. In the general case, the different subbands will have different sensitivities due to effects such as a different amount of data flagged due to RFI and frequency variations of receiver and sky noise. Thus, the combination of subband images needs to be a weighted average.

All else being equal, weighting by the inverse subband variance ($1/\sigma^2$) gives the optimal result. However, at low frequencies where the disk of the Milky Way is very bright and has a very steep spectrum, a $1/\sigma^2$ weighting can effectively remove the effects of the lowest frequencies from images at low Galactic latitude. In this case, $1/\sigma$ weighting may be preferred.

When it is possible to fit for a meaningful in–band spectral index, it is desirable to do so. A weighted least squares fit subject to signal–to–noise constraints can produce spectral index values for some pixels, others should be blanked - given a value meaning "no value". The technique described here is also discussed in [2] and [3].

National Radio Astronomy Observatory, 520 Edgemont Rd., Charlottesville, VA, 22903 USA email: bcotton@nrao.edu

[1]http://www.cv.nrao.edu/∼bcotton/Obit.html

III. Obit Implementation

In the following, it is assumed that the broadband image(s) are derived from a number of subbands in frequency that are to be combined by a weighted average ($I_{total}(x,y)$):

$$I_{total}(x,y) = \frac{\sum_{i=1}^{n} W_i I_i(x,y)}{\sum_{i=1}^{n} W_i}, \quad (1)$$

where i of n is the subband index, $I_i(x,y)$ is the image of subband i and W_i is the weight of subband i. The effective frequency (ν_{eff}) of this combination will be

$$\nu_{eff} = \frac{\sum_{i=1}^{n} \nu_i W_i}{\sum_{i=1}^{n} W_i}, \quad (2)$$

where ν_i is the effective frequency of subband i. If multiple such images are to be produced using the same reference frequency, then all should use the same channelization (subbands) and associated weights. The following sections describe how to implement frequency averaging to maintain a constant effective frequency. This is performed using a set of python scripts to be executed from ObitTalk. These files are kept in $OBIT/share/scripts and should be copied to your working directory to be edited and renamed for the details of your project. These scripts assume input images in the form produced by Obit task MFImage [4]. All input and output images are in FITS image files of the type produced by Obit or AIPS.

A. Determining Weights

As a general rule, some measure of the "noise" in each subband should be used to set the relative weight of that subband. The optimal weighting in a statistical sense is by the inverse variance (σ_i^{-2}) of each subband. However, in some cases, e.g. at low Galactic latitudes, the very steep spectrum of the additional sky noise causes this weighting to effectively discard the lower portion of the band. Weighting by the inverse of σ_i (subband RMS) will preserve more of the lower frequency information but still give higher weight to less noisy data.

If only one image is to be produced with a given effective frequency, the weights can be determined from the relevant subband images. If multiple images are to be produced, e.g. extended area mosaics or a set of related pointings, then the appropriate weighting need be determined from an average over all relevant images. Figures 1 & 2 give the text of script $OBIT/share/scripts/AvgWeights.py which operates on one or more broadband image cubes of the type produced by Obit task MFImage [4] and determines the weights to be be used for the images. The results are stored in the pickle file AvgWeights.pickle for use by other scripts. The relevant parameters set in this script are:

- **dir.** This is the relative path to the directory containing the FITS images. For the CWD this is './'
- **posn.** This is a list of the roots of the file names.
- **post.** This is the constant part of the file names. The full name of the files is dir+posn[i]+post for each posn[i].
- **doLin.** True if the weighting is $1/\sigma$, else $1/\sigma 2$
- **nthreads.** The number of threads to use; generally the number of cores available.

The script can be executed from ObitTalk by:

```
>>> exec(open('AvgWeights.py').read())
```

When completed, the script will give the effective frequency. None of the input files are modified. The script should be renamed as appropriate for your project.

B. Fitting Spectra in MFImage-like Cubes

Once the weights have been determined, the values stored in the AvgWeight.pickle file can be used to fit spectra to be filled into the first one or two planes on the MFImage-like file(s). There are a number of options which are set by parameters in the script. A nonlinear least squares fit is attempted in each pixel for which the broadband total intensity exceeds *minFlux* and for which adding the spectral index term does not significantly increase the χ^2 of the fit. The spectral index values fitted are given in the second plane of the image and blanked if not fitted The broadband flux densities are always the weighted average even if the fitting produced a spectral index. Primary beam corrections are optional but should NOT be used in mosaiced images for which they have already been applied.

The spectral fitting will use all valid pixels in the cube. If this is a single pointing cube (i.e. not a mosaic) and primary beam corrections have been applied with a cutoff (*PBmin*) then the values at this top of the band may not be included which will result in a lower effective frequency for affected pixels. To counteract this, the *doBlank=True* option will blank the first two planes on the output image where the highest frequency bin is blanked; thus ensuring a constant effective frequency.

By default the files to be operated on are those used to derive the weights but a different set, defined by *post* and *fitsfiles* can be specified. The relevant parameters set in this script are:

- **minFlux.** Minimum flux density for fitting spectral index.
- **defaultSI.** Default initial spectral index in fit.
- **doPBCor.** Apply primary beam correction in determining broadband flux density and spectral index? The output subband images will not be affected.
- **antSize.** The antenna diameter (m) to be used in primary beam corrections. MeerKAT images produced in Obit are recognized and a cosine squared beam is used.
- **PBmin.** Minimum primary beam gain to be corrected; pixels with lower values will be blanked.
- **doBlank.** Blank broadband image and spectral index where the highest frequency subband is blanked, or, would be blanked by a primary beam correction.
- **nthreads.** Number of threads to use in computations.
- **post.** The equivalent of post described in Section III-A; if not given then the value saved in the AvgWeight.pickle file is used.
- **fitsfiles.** A list of (File_Name_Root, image_RMS(not used)) for the input FITS files. The full path of the files used are dir+fitsfile[i][0]+post. If not given, the posn list from the AvgWeight.pickle file is used.

The script is given in Figures 3 & 4 and can be executed from ObitTalk by:

```
>>> exec(open('FitSpec.py').read())
```

On successful completion of the script, the first two planes of the output image(s) will be replaced by the average broadband image and spectral index plane (if fitted) and the reference frequency will be changed to the effective frequency. Subband planes are not modified. The script should be renamed as appropriate for your project.

C. Spectral Fit Plus Errors

An alternate, more compact, rendition of the image with just the fitted spectrum together with the least squares error estimates is also possible. A new output FITS file(s) is(are) produced. Weighting is obtained from the AvgWeight.pickle file. The relevant parameters set in this script are similar to those in Section III-B:

- **minFlux.** Minimum flux density for fitting spectral index.
- **defaultSI.** Default initial spectral index in fit.
- **doPBCor.** Apply primary beam correction in determining broadband flux density and spectral index?
- **antSize.** The antenna diameter (m) to be used in primary beam corrections. MeerKAT images produced in Obit are recognized and a cosine squared beam is used.
- **PBmin.** Minimum primary beam gain to be corrected; pixels with lower values will be blanked.
- **maxChi2.** Fit acceptance level
- **calFract.** % relative calibration error for error estimates.
- **doBlank.** Blank broadband image and spectral index where the highest frequency subband is blanked, or, would be blanked by a primary beam correction.
- **nthreads.** Number of threads to use in computations.
- **post.** The equivalent of post described in Section III-A; if not given then the value saved in the AvgWeight.pickle file is used.
- **fitsfiles.** A list of (File_Name_Root, image_RMS(not used) for the input FITS files. The full path of the files used are dir+fitsfile[i][0]+post. If not given, the posn list from the AvgWeight.pickle file is used.
- **opost.** The final part of the output FITS file name, default "_5Pln.fits". The output FITS files will have names dir+fitsfile[i][0]+opost.

The script is given in Figures 5 & 6 can be executed from ObitTalk by:

```
>>> exec(open('FitError.py').read())
```

On successful completion of the script, a new FITS image(s) will be generated from each input image cube with planes 1) broadband image (in Jy), 2) spectral index where fitted else blanked, 3) uncertainty of plane 1) 4) uncertainty of plane 2 where fitted, 5) reduced χ^2 of fit. The reference frequency will be the effective frequency. The input images are not modified. The script should be renamed as appropriate for your project.

Fig. 1. AvgWeights.py

```
# average Stokes I channel weights in a set of images
#exec(open('AvgWeights.py').read())
# Saves information in dict saved in AvgWeights.pickle
# weights, one per subband, RMSes:[mosaic_name,RMS],
# Freqs:[center freq, MHz], EffFreq: Effective frequency,
# dir:data directory, post: ending of FITS file name
print("\nDetermining weighting, effective frequency")

# Assumes MFImage output
dir = "../data/"      # Data directory
# file name roots
posn = ["Field_1" ,"Field_2", "Field_3", "Field_4"]
post = '_I_Mosaic.fits' # file=dir+<name root>+post
doLin = True  # If True weight by 1/sigma, else 1/sigma^2
nthreads = 16  # Number of threads to use

import Image, OErr, OSystem, FArray
OSystem.PAllowThreads(nthreads)  # with threads
from PipeUtil import SaveObject, FetchObject
import math, pickle

# Function to get RMSHist for plane planeno
def imrms(inImage, planeno=1):
    """ Get plane RMSHist
    Returns dictionary with statistics of selected region with entries:
    * inImage   = Python Image object, created with getname, getFITS
    """
    inImage.GetPlane(None,[planeno,1,1,1,1],err)
    rms = inImage.FArray.RMS
    return rms
# end imrms

# Get information from first file
x=getFITS(dir+posn[0]+post,0)
# test that MFImage output
if x.Desc.Dict['ctype'][2]!='SPECLNMF':
    raise RuntimeError("not MFImage output")

nterm  = x.Desc.List.Dict['NTERM'][2][0]
nspec  = x.Desc.List.Dict['NSPEC'][2][0]
nplane = x.Desc.Dict['inaxes'][2]

# Get frequencies
freqs = []
for i in range(1,nspec+1):
    key = "FREQ%4.4d"%i
    freqs.append(x.Desc.List.Dict[key][2][0]*1.0e-6)

# end channel loop

# Channel RMSes per image stored in result[p]
result = {}
for i in range (0,len(posn)):
    p=posn[i]
    x=getFITS(dir+p+post,0)
    print ("File",dir+p+post)
    rms = []
```

Fig. 2. AvgWeights.py cont'd

```
    for j in range(nterm+1,nplane+1):
        s = imrms(x,j)
        rms.append(s)

    # end channel loop
    result[p] = rmsx
# end image loop

# broadband rmses
ff = []
for i in range (0,len(posn)):
    p=posn[i];  x=getFITS(dir+p+post,0)
    s = imrms(x, 1)
    ff.append((p,s))

# Average weights
for i in range (0,len(posn)):
    p=posn[i];  wt = []; rms = result[p]
    for j in range(0,len(rms)):
        if rms[j]>0.0:
            if doLin:
                wt.append(1.0e-6*(rms[j]**-1))  # weight by 1/sigma
            else:
                wt.append(1.0e-6*(rms[j]**-2)) # weight by 1/sigma^2
        else:
            wt.append(0.0)

# end channel loop
# Normalize by sum of weights
sumwt = sum(wt)
for i in range(0,len(wt)):
    wt[i] = 100*wt[i]/sumwt

# Print subband weights
print ("Subband  Freq   Weight")
for i in range(0,len(wt)):
    print ("%3d %9.2f %8.2f"%(i+1,freqs[i],wt[i]))

# Effective frequency
sum1 = 0.0; sumwt=0.0
for i in range(0,len(wt)):
    sum1 += freqs[i]*wt[i]; sumwt += wt[i];

EffFreq = sum1/sumwt
print ("\nEffective Frequency = %9.2f"%EffFreq,"MHz")

# Save to pickle jar
# weights, one per subband, RMSes:[mosaic_name,RMS], Freqs:[center freq, MHz],
# dir:data directory, post: ending of FITS file name
stuff = {"weights":wt, "RMSes":ff, "Freqs":freqs, "EffFreq":EffFreq, \
         "dir":dir, "post":post}
SaveObject(stuff, "AvgWeights.pickle", True)
print ("Wrote AvgWeights.pickle" )
```

Fig. 3. FitSpec.py

```
# Fit spectra to a set of images where flux density>minFlux
# Using weighting from AvgWeights.pickle
#exec(open('FitSpec.py').read())
minFlux   = 0.0002 # Minimum flux density for SI
defaultSI = 0.0    # Default spectral index
doPBCor   = False  # Do PB correct? not for Mosaics
antSize   = 24.5   # Antenna size for PB Corr
PBmin     = 0.05   # Min PB gain for correction
doBlank   = False  # Blank total intensity and SI where highest plane blanked?
                   # After any PB Correction
nthreads  = 16     # Number of threads to use
post      = None   # If given, the ending of file names,
                   # Else the value from AvgWeights.pickle
fitsfiles = None   # If given, (name root, broadband RMS)
                   # Else the value from AvgWeights.pickle
import Image, OErr, OSystem, FArray
OSystem.PAllowThreads(nthreads)  # with threads
from PipeUtil import SaveObject, FetchObject
import math, pickle

# Get weighting info
stuff = FetchObject("AvgWeights.pickle")
refFreq   = stuff['EffFreq'] # Effective frequency in MHz
wwts      = stuff['weights'] # Subband weights in %
dir       = stuff['dir']     # Data directory
if fitsfiles==None:
    fitsfiles = stuff['RMSes']   # List of (file_name root, RMS)
if post==None:
    post = stuff['post']    # Ending of file name

import Obit, ImageMF, Image, FArray, OSystem, OErr
err = OErr.OErr()
OSystem.PAllowThreads(nthreads)  # with threads

for fitsfile in fitsfiles:
    print ("Doing",fitsfile[0])
    inMF  = ImageMF.newPFImageMF('Input',dir+fitsfile[0]+post, 0, True, err)
    ImageMF.PFitSpec(inMF, err, antSize=0., nOrder=1,corAlpha=defaultSI,
                     refFreq=refFreq*1.0e6, Weights=wwts, minFlux=minFlux,
                     doPBCor=doPBCor, PBmin=PBmin)
```

IV. Discussion

This memo discusses producing a set of broadband continuum images producing a constant and well defined effective frequency. Scripts are provided which can be used in the Obit environment to produce such images from an input set of single pointing or mosaic images produced in Obit.

References

[1] W. D. Cotton, "Obit: A Development Environment for Astronomical Algorithms," *PASP*, vol. 120, pp. 439–448, 2008.

[2] S. Goedhart, W. D. Cotton, F. Camilo, M. A. Thompson, G. Umana, M. Bietenholz, P. A. Woudt, L. D. Anderson, C. Bordiu, D. A. H. Buckley, C. S. Buemi, F. Bufano, F. Cavallaro, H. Chen, J. O. Chibueze, D. Egbo, B. S. Frank, M. G. Hoare, A. Ingallinera, T. Irabor, R. C. Kraan-Korteweg, S. Kurapati, P. Leto, S. Loru, M. Mutale, W. O. Obonyo, A. Plavin, S. H. A. Rajohnson, A. Rigby, S. Riggi, M. Seidu, P. Serra, B. M. Smart, B. W. Stappers, N. Steyn, M. Surnis, C. Trigilio, G. M. Williams, T. D. Abbott, R. M. Adam, K. M. B. Asad, T. Baloyi, E. F. Bauermeister, T. G. H. Bennet, H. Bester, A. G. Botha, L. R. S. Brederode, S. Buchner, J. P. Burger, T. Cheetham, K. Cloete, M. S. de Villiers, D. I. L. de Villiers, L. J. du Toit, S. W. P. Esterhuyse, B. L. Fanaroff, D. J. Fourie, R. R. G. Gamatham, T. G. Gatsi, M. Geyer, M. Gouws, S. C. Gumede, I. Heywood, A. Hokwana, S. W. Hoosen, D. M. Horn, L. M. G. Horrell, B. V. Hugo, A. I. Isaacson, G. I. G. Józsa, J. L. Jonas, J. D. B. L. Jordaan, A. F. Joubert, R. P. M. Julie, F. B. Kapp, N. Kriek, H. Kriel, V. K. Krishnan, T. W. Kusel, L. S. Legodi, R. Lehmensiek, R. T. Lord, P. S. Macfarlane, L. G. Magnus, C. Magozore, J. P. L. Main, J. A. Malan, J. R. Manley, S. J. Marais, M. D. J. Maree, A. Martens, P. Maruping, K. McAlpine, B. C. Merry, M. Mgodeli, R. P. Millenaar, O. J. Mokone, T. E. Monama, W. S. New, B. Ngcebetsha, K. J. Ngoasheng, G. D. Nicolson, M. T. Ockards, N. Oozeer, S. S. Passmoor, A. A. Patel, A. Peens-Hough, S. J. Perkins, A. J. T. Ramaila, S. M. Ratcliffe, R. Renil, L. L. Richter, S. Salie, N. Sambu, C. T. G. Schollar, L. C. Schwardt, R. L. Schwartz, M. Serylak, R. Siebrits, S. K. Sirothia, M. J. Slabber, O. M. Smirnov, A. J. Tiplady, T. J. van Balla, A. van der Byl, V. Van Tonder, A. J. Venter, M. Venter,

Fig. 4. FitSpec.py cont'd

```
    if doBlank:
        print ('Blanking by high freq. bin')
        # Blanking mask
        outIm = Image.newPFImage('Output',dir+fitsfile[0]+post, 0, True, err)
        hiPlane = [outIm.Desc.Dict['inaxes'][2],1,1,1,1]
        if doPBCor:
            # Mask by primary beam correction and high bin
            PBImg = Image.Image('PBImage'); Image.PCloneMem(outIm,PBImg,err)
            # Blank where highest plane blanked
            ImageUtil.PPBImage(outIm,PBImg,err,PBmin,outPlane=hiPlane,antSize=antSize)
            PBImg.GetPlane(None,hiPlane, err); mask1 = PBImg.FArray
            outIm.GetPlane(None,hiPlane, err); FArray.PBlank(mask1, outIm.FArray, mask1)
        else:
            # Mask by highest frequency
            outIm.GetPlane(None,hiPlane, err); mask1 = FArray.PCopy(outIm.FArray, err)
            PBImg = None
        # Plane 1 - total intensity
        outIm.GetPlane(None, [1,1,1,1,1], err)
        FArray.PBlank(outIm.FArray, mask1, outIm.FArray)
        outIm.PutPlane(None, [1,1,1,1,1], err)
        # Plane 2 - Spectral index
        outIm.GetPlane(None, [2,1,1,1,1], err)
        FArray.PBlank(outIm.FArray, mask1, outIm.FArray)
        outIm.PutPlane(None, [2,1,1,1,1], err)
        #DEBUG Image.PFArray2FITS(mask1, "Mask.fits", err, outDisk=0)
        del PBImg, mask1
# End blanking
# end loop
```

M. G. Welz, and L. P. Williams, "The SARAO MeerKAT 1.3 GHz Galactic Plane Survey," *MNRAS*, vol. 531, no. 1, pp. 649–681, Jun. 2024.

[3] W. D. Cotton, M. D. Filipović, F. Camilo, R. Indebetouw, R. Z. E. Alsaberi, J. O. Anih, M. Baker, T. S. Bastian, I. Bojičić, E. Carli, F. Cavallaro, E. J. Crawford, S. Dai, F. Haberl, L. Levin, K. Luken, C. M. Pennock, N. Rajabpour, B. W. Stappers, J. T. van Loon, A. A. Zijlstra, S. Buchner, M. Geyer, S. Goedhart, and M. Serylak, "The MeerKAT 1.3 GHz Survey of the Small Magellanic Cloud," *MNRAS*, vol. 529, no. 3, pp. 2443–2472, Apr. 2024.

[4] W. D. Cotton, "Wide–band, Wide–field Imager MFImage," *Obit Memo series*, vol. 63, pp. 1–2, 2019. [Online]. Available: https://www.cv.nrao.edu/ bcotton/ObitDoc/MFImage.pdf

Fig. 5. FitError.py

```
# Fit spectra with errors to a set of images where flux density>minFlux
# Writes new 5 plane cube
# Using weighting from AvgWeights.pickle
#exec(open('FitError.py').read())
minFlux   = 0.0002 # Minimum flux density for SI
defaultSI = 0.0    # Default spectral index
doPBCor   = False  # Do PB correct? not for Mosaics
maxChi2   = 0.01   # fit acceptance level
calFract  = 0.00   # % relative calibration error for error estimates
antSize   = 24.5   # Antenna size for PB Corr
PBmin     = 0.05   # Min PB gain for correction
doBlank   = False  # Blank total intensity and SI where highest plane blanked?
                   # After any PB Correction
nthreads  = 16     # Number of threads to use
post      = None   # If given, the ending of file names,
                   # Else the value from AvgWeights.pickle
fitsfiles = None   # If given, fine name root, RMS
                   # Else the value from AvgWeights.pickle
opost     = "_5Pln.fits" # Post for output
doBlank   = True  # Blank total intensity and SI where highest plane blanked?

import Image, OErr, OSystem, FArray
OSystem.PAllowThreads(nthreads)  # with threads
from PipeUtil import SaveObject, FetchObject
import math, pickle

# Get weighting info
stuff = FetchObject("AvgWeights.pickle")
refFreq   = stuff['EffFreq'] # Effective frequency in MHz
wwts      = stuff['weights'] # Subband weights in %
dir       = stuff['dir']     # Data directory
if fitsfiles==None:
    fitsfiles = stuff['RMSes']   # List of (file_name root, RMS)
if post==None:
    post = stuff['post']    # Ending of spectral cube file name

import Obit, ImageMF, Image, FArray, OSystem, OErr
err = OErr.OErr()
OSystem.PAllowThreads(nthreads)  # with threads

for fitsfile in fitsfiles:
    print ("Doing",fitsfile[0])
    inMF   = ImageMF.newPFImageMF('Input',dir+fitsfile[0]+post, 0, True, err)
    outMF  = ImageMF.newPFImageMF('Output',dir+fitsfile[0]+opost, 0, False, err)
    ImageMF.PFitSpec2(inMF, outMF, err, \
                      corAlpha=defaultSI, doError=True, maxChi2=maxChi2,calFract=calFract, \
                      refFreq=refFreq*1.0e6, Weights=wwts, minFlux=minFlux, doPBCor=doPBCor)
```

Fig. 6. FitError.py cont'd

```
    if doBlank:
        print ('Blanking by high freq. bin')
        # Blanking mask
        outIm = Image.newPFImage('Output',dir+fitsfile[0]+post, 0, True, err)
        hiPlane = [outIm.Desc.Dict['inaxes'][2],1,1,1,1]
        if doPBCor:
            # Mask by primary beam correction and high bin
            PBImg = Image.Image('PBImage'); Image.PCloneMem(outIm,PBImg,err)
            ImageUtil.PPBImage(outIm,PBImg,err,PBmin,outPlane=hiPlane,antSize=antSize)
            PBImg.GetPlane(None,hiPlane, err); mask1 = PBImg.FArray
            outIm.GetPlane(None,hiPlane, err); FArray.PBlank(mask1, outIm.FArray, mask1)
        else:
            # Mask by highest frequency
            outIm.GetPlane(None,hiPlane, err); mask1 = FArray.PCopy(outIm.FArray, err)
            PBImg = None
        # Plane 1 - total intensity
        outIm  = Image.newPFImage('Output',dir+fitsfile[0]+opost, 0, True, err)
        outIm.GetPlane(None, [1,1,1,1,1], err)
        FArray.PBlank(outIm.FArray, mask1, outIm.FArray)
        outIm.PutPlane(None, [1,1,1,1,1], err)
        # Plane 2 - Spectral index
        outIm.GetPlane(None, [2,1,1,1,1], err)
        FArray.PBlank(outIm.FArray, mask1, outIm.FArray)
        outIm.PutPlane(None, [2,1,1,1,1], err)
        Image.PFArray2FITS(mask1, "Mask.fits", err, outDisk=0)
        del PBImg
# end loop
```