Chapter 1

General

package for general use astronomical processing

include ”general.g”

Description
The general package contains modules that are of general use for astronomical processing.

1.1 deconvolver - Module

Module for deconvolving images
include ”deconvolver.g”

Description

`deconvolver` provides a unified interface for image processing including deconvolution starting from images.

Tools

deconvolver	deconvolver tool

1.1.1 deconvolver - Tool

Package	**general**
Module	**deconvolver**

deconvolver tool

Description
deconvolver is a tool that deconvolves a known point spread function from an image. A `deconvolver` must be constructed for each dirty image and point spread function for which one wishes to do processing. Multicple copies of `deconvolver` may be made at any time (provide they are given different names).

Constructors

deconvolver	Construct a deconvolver tool
deconvolvertester	Construct a deconvolver tool for testing

Functions

boxmask	Construct a mask from blc, trc
clarkclean	Make a clean image using the Clark Clean a threshold
clean	Make a clean image with Hogbom or MultiScale Clean
clipimage	Zero all pixels where Stokes I is below a threshold
close	Close the deconvolver tool
convolve	Convolves an image with the PSF
dirtyname	Return the name of the dirty-image table
done	Terminate the deconvolver process
ft	Fourier transform the specified model
id	Return the Glish ID of the tool
make	Make an empty image
makegaussian	Make an image with a single gaussian component
makeprior	Make the mem's prior image, or make a mask
mem	Make the mem image
open	Open a new dirty image and PSF
pixon	Deconvolve using the Pixon algorithm
psfname	Return the name of the PSF-image table
reopen	Reopen the dirty image and PSF
residual	Find the residuals
restore	Restore the residuals
setscales	Set the scale sizes for MultiScale Clean
smooth	smooth the image
state	Return the "state" of the tool
summary	Summarize the current state
type	Return the type of the of the tool
updatestate	[A GUI builders related function]Update the GUI to reflect the currect state

deconvolver - Constructor

Package	**general**
Module	**deconvolver**
Tool	**deconvolver**

Construct a deconvolver tool

Synopsis
deconvolver(dirtyname, psfname)

Description
This is used to construct `deconvolver` tools associated with a dirty image and point spread function. The `deconvolver` tool may then be used to generate various types of images. Note that a new executable is started every time the constructor is called.
This returns a Glish variable containing the tool functions of deconvolver.

Arguments

dirtyname	Dirty image to be processed Allowed:Table name Default: 'dirty'
psfname	point spread function to be processed Allowed:Table name Default: 'psf'

Returns
deconvolver tool

Example

```
deco:=deconvolver('3C273XC1.dirty', '3C273XC1.psf')
deco.clean(model='3C273XC1.clean', niter=10000, gain=0.2)
deco.close()
```

---

deconvolvertester - Constructor

Package	**general**
Module	**deconvolver**
Tool	**deconvolver**

Construct a deconvolver tool for testing

Synopsis
deconvolvertester(filename, size, stokes, coordinates)

Description
This creates a `deconvolver` tool that one can use for testing. A standard MeasurementSet is created and then used to initialize imager, which in turn is used to generate a dirty image and PSF. The dataset used is an observation of 3C273 at 8GHz using the C-configuration of the VLA. It is part of the AIPS++ distribution.

Arguments

filename	Name of MS to create Allowed:String Default: 3C273XC1.ms
size	Number of pixels on a side Allowed:Int Default: 128
stokes	Stokes parameters to image (e.g. 'IQUV') Allowed:'I'—'IV'—'IQU'—'IQUV' Default: 'I'
coordinates	Coordinate system to use e.g. 'j2000,'gal' Allowed:String Default: 'j2000'

Returns
deconvolver tool

Example

```
deco:=deconvolvertester(coordinates='gal',size=1024)
deco.clean(model='3C273XC1.clean');
deco.close()
```

---

deconvolver.open - Function

Package	**general**
Module	**deconvolver**
Tool	**deconvolver**

Open a new dirty image and PSF

Synopsis
open(dirty, psf)

Description
Close the current images and open a new dirty image and PSF instead. The current state of `deconvolver` is retained, except for the data selection.

Arguments

dirty	Dirty image to be processed Allowed:Table name Default: 'dirty'
psf	point spread function to be processed Allowed:Table name Default: 'psf'

Returns
Bool

---

deconvolver.reopen - Function

Package	**general**
Module	**deconvolver**
Tool	**deconvolver**

Reopen the dirty image and PSF

Synopsis
reopen()

Description
Close and reopen the current dirty and PSF images, and make new convolvers and cleaners. The main benefit of this method is to flush the residual image and replace it with the dirty image.

Arguments

Returns
Bool

---

deconvolver.close - Function

Package	**general**
Module	**deconvolver**
Tool	**deconvolver**

Close the deconvolver tool

Synopsis
close()

Description
This is used to close `deconvolver` tools. Note that the data is written to disk. The `deconvolver` process keeps running until a done tool function call is performed.

Returns
Bool

Example

```
deco:=deconvolver('3C273XC1.dirty', '3C273XC1.psf')
deco.clean(model='3C273XC1.clean');
deco.close()
```

---

deconvolver.done - Function

Package	**general**
Module	**deconvolver**
Tool	**deconvolver**

Terminate the deconvolver process

Synopsis
done()

Description
This is used to totally stop the `deconvolver` process. It is a good idea to conserve memory use on your machine by stopping the process once you no longer need it.

Returns
Bool

Example

```
deco:=deconvolver('3C273XC1.dirty', '3C273XC1.psf')
deco.clean(model='3C273XC1.clean');
deco.close()
deco.done()
```

---

deconvolver.summary - Function

Package	**general**
Module	**deconvolver**
Tool	**deconvolver**

Summarize the current state

Synopsis
summary()

Description
Writes a summary of the properties of the deconvolver to the default logger. This includes:

- The names of the dirty image and PSF (set in construction or via the open function.

- The current beam fit

Returns
Bool

Example

```
deco:=deconvolver('3C273XC1.dirty', '3C273XC1.psf')
deco.summary()
```

---

deconvolver.boxmask - Function

Package	**general**
Module	**deconvolver**
Tool	**deconvolver**

Construct a mask from blc, trc

Synopsis
boxmask(mask, blc, trc, value)

Description
A mask image is an image with the same shape as the other images but with values between 0.0 and 1.0 as a pixel value. Mask images are used in deconvolver to control the region selected in a deconvolution.

In the Clark CLEAN, the mask image can usefully have any value between 0.0 and 1.0. Intermediate value discourage but do not rule out selection of clean components in that region. This is accomplished by multiplying the residual image by the mask prior to entering the minor cycle. Note that if you do use a mask for the Clark or Hogbom Clean, it must cover only a quarter of the image. boxmask does not enforce this requirement.

Arguments

mask	name of mask image Allowed:String
blc	Bottom left corner Allowed:Vector Default: [1,1,1,1]
trc	Top right corner Allowed:Vector Default: image shape
value	Value to fill in Allowed:Float Default: 1.0

Returns
Bool

Example

```
deco.boxmask(mask='bigmask', blc=[56,45,1,1], trc=[87,93,4,1])
deco.clean(mask='bigmask', model='3C273XC1.clean.masked', niter=1000)
```

Makes the image bigmask, and then sets it to unity for all points in the region bounded by the blc and trc. Then cleans using it as the mask.

---

deconvolver.clipimage - Function

Package	**general**
Module	**deconvolver**
Tool	**deconvolver**

Zero all pixels where Stokes I is below a threshold

Synopsis
clipimage(image, threshold)

Description
All pixels in the image with Stokes I less than some threshold are set to zero. This is useful prior to self-calibration where one oftens wishes to remove negative pixels from the model. Note that if the image has polarization information, then the polarized part of a pixel is also set to zero if Stokes I is less than the threshold.

Arguments

image	name of image Allowed:String	
threshold	Threshold Allowed:String Default: '0Jy'	

Returns
Bool

Example

```
imgr.clipimage(image='clean', threshold='50mJy')
imgr.selfcal(cal='mycal', model='clean');
```

---

deconvolver.clarkclean - Function

Package	**general**
Module	**deconvolver**
Tool	**deconvolver**

Make a clean image using the Clark Clean a threshold

Synopsis
clarkclean(niter, gain, threshold, displayprogress, model, mask, histbins, psfpatchsize, maxextpsf, speedup, maxnumpix, maxnummajcycles, maxnummineriter)

Description

In the Clark Clean algorithm, the cleaning is split into minor and major cycles. In the minor cycles only the brightest points are cleaned, using a subset of the point spread function. In the major cycle, the points thus found are subtracted correctly by using an FFT-based convolution.

Arguments

niter	Number of iterations Allowed:Int Default: 1000
gain	Loop Gain for CLEANing Allowed:Double Default: 0.1
threshold	Flux level at which to stop CLEANing Allowed:Quantity Default: '0Jy'
displayprogress	Display the progress of the cleaning? Allowed:Bool Default: F
model	Name of images Allowed:String
mask	Name of mask images used for CLEANing Allowed:String
histbins	Number of bins in the pixel-flux histogram Allowed:Int Default: 500
psfpatchsize	Size of PSF for minor cycle Allowed:Vector of Int Default: [51,51]
maxextpsf	maximum external sidelobe, used to set depth of minor cycle clean Allowed:Double Default: 0.2
speedup	Cleaning speedup exponent Allowed:Int Default: 0
maxnumpix	Maximum number of pixels used in each minor cycle Allowed:Int Default: 10000
maxnummajcycles	Max number of major cycles; -1 = no restrictions Allowed:Int Default: -1
maxnummineriter	Max number of minor iterations; -1 = no restrictions Allowed:Int Default: -1

Returns
Bool

---

deconvolver.dirtyname - Function

Package	**general**
Module	**deconvolver**
Tool	**deconvolver**

Return the name of the dirty-image table

Synopsis
dirtyname()

---

deconvolver.psfname - Function

Package	**general**
Module	**deconvolver**
Tool	**deconvolver**

Return the name of the PSF-image table

Synopsis
psfname()

---

deconvolver.make - Function

Package	**general**
Module	**deconvolver**
Tool	**deconvolver**

Make an empty image

Synopsis
make(image, async)

Description
Make an empty image with the properties (co-ordinate system etc.) borrowed from the dirty image.

Arguments

image	Name of the new image on the disk Allowed:String Default: ”
async	Run asynchronously in the background? Allowed:Bool Default: !dowait

---

deconvolver.convolve - Function

Package	**general**
Module	**deconvolver**
Tool	**deconvolver**

Convolves an image with the PSF

Synopsis
convolve(convolvemodel, model)

Description
Convolves an image (e.g., the model image) with the PSF

Arguments

convolvemodel	Name of the output image on the disk to hold the result of the convolution Allowed:String Default: ”
model	The input image to be convolved with the PSF Allowed:String Default: ”

---

deconvolver.makegaussian - Function

Package	**general**
Module	**deconvolver**
Tool	**deconvolver**

Make an image with a single gaussian component

Synopsis
makegaussian(gaussianimage, bmaj, bmin, bpa, normalize, async)

Description
Make a model image with the a single gaussian. The properties of the output image (e.g. the co-ordinate system, etc.) are borrowed from the dirty image. The image is made as follows:
I(x,y)= Delta function of unit amplitude at (0,0)
Temp(x,y) = Gaussian(x,y,Amplitude, Center, Sigma, PA)
I(x,y) = Convovlution of Temp(x,y) with I(x,y).
If `normalize=T` I(x,y) = I(x,y)/(area under the gaussian).

Arguments

gaussianimage	Name of the output image on the disk Allowed:String Default: ”
bmaj	The major axis of the gaussian Allowed:String Default: ’0rad’
bmin	The minor axis of the gaussian Allowed:String Default: ’0rad’
bpa	The Position Angle of the gaussian Allowed:String Default: ’0deg’
normalize	Normalize the area under the gaussian to 1.0? Allowed:Bool Default: T
async	Run asynchronously in the background? Allowed:Bool Default: !dowait

---

deconvolver.type - Function

Package	**general**
Module	**deconvolver**
Tool	**deconvolver**

Return the type of the of the tool

Synopsis
type()

Description
Always returns the string "deconvolver".

---

deconvolver.id - Function

Package	**general**
Module	**deconvolver**
Tool	**deconvolver**

Return the Glish ID of the tool

Synopsis
id()

Description

---

deconvolver.state - Function

Package	**general**
Module	**deconvolver**
Tool	**deconvolver**

Return the "state" of the tool

Synopsis
state()

Description
Prints the name of the Dirty Image and the PSF and the parameters of the gaussian fitted to the main lobe of the PSF (the "Clean Beam").

---

deconvolver.updatestate - Function

Package	**general**
Module	**deconvolver**
Tool	**deconvolver**

[A GUI builders related function]Update the GUI to reflect the currect state

Synopsis
updatestate(f, method)

Description
Updates the GUI to reflect the currect state of the tool. This function is used by toolmanager. See documentation of the toolmanager for details about "methods" used to update the GUI.

Arguments

f	Glish variable for the GUI to be updated Default: "
method	The method to be used for updating Allowed:'INIT','DONE',or 'close' Default: 'INIT'

---

deconvolver.pixon - Function

Package	**general**
Module	**deconvolver**
Tool	**deconvolver**

Deconvolve using the Pixon algorithm

Synopsis
pixon(sigma, model, imageplane, async)

Description
Makes a image using the Pixon algorithm. According to its developers, the pixon method is:

> a new way of viewing the problem of modeling the underlying, unblurred, noise-free image. The goal of the [new] Pixon method

> model was to construct the simplest, i.e. smoothest, model for the image that would be consistent with the data, i.e. have an acceptable chi-square fit. Being the simplest model, the derived image would be artifact free, i.e. there would be no spurious sources, since by construction the simplest model eliminates unneeded structures. In addition, the model would necessarily be a "critical" model, i.e. most tightly constrained by the data, and consequently have the most accurately determined parameters.

In the simplest terms, the pixon method smooths a model locally as much as is allowed by the specified noise level. This, like all high performance estimation methods, the pixon approach works best when the noise level is known and well-characterized.
The Pixon algorithm is available via an IDL library courtesy of the Pixon LLC. This means that you must have the library and IDL installed on your computer. The Pixon library is available free of charge for your personal scientific use direct from Pixon LLC. IDL is available commercially from RSI.

Arguments

sigma	Noise level to try to achieve Allowed:String Default: '0.001Jy'
model	Name of image Allowed:String
imageplane	Is this an image plane problem (like single dish)? Allowed:Bool Default: F
async	Run asynchronously in the background? Allowed:Bool Default: !dowait

Returns
Bool

---

deconvolver.clean - Function

Package	**general**
Module	**deconvolver**
Tool	**deconvolver**

Make a clean image with Hogbom or MultiScale Clean

Synopsis
clean(algorithm, niter, gain, threshold, model, mask, async)

Description

Makes a clean image using either the Hogbom or MultiScale algorithms. The MultiScale algorithm is the default. The clean is performed on the residual image calculated from the dirty image minus the point spread function convolved with the model currently selected. Thus if you want to restart a clean, simply set the model to the model that was previously produced by clean.
Rather than explicit CLEAN boxes, mask images are used to constrain the region that is to be deconvolved. To make mask images, use either boxmask (to define a mask via the corner locations blc and trc) or mask (to define a mask via thresholding an existing image). The default mask is the inner quarter of the image.
The CLEAN deconvolution is joint in whatever Stokes parameters are present. Thus it searchs for peaks in either I or $I + |V|$ or $I + \sqrt{Q^2 + U^2 + V^2}$, the rationale for the latter two forms being to be biased towards finding strongly polarized pixels first (these forms are also the maximum eigenvalue of the coherency matrix). The PSF is constrained to be the same in all polarizations (a feature of this implementation, not of the Hamaker-Bregman-Sault formalism).
The clean algorithms possible are:

Hogbom The classic algorithm: delta function units of emission are found iteratively by searching for the peak. Each point is subtracted from the full residual image using the shifted and scaled point spread function.

Multi-Scale As the Multi-Scale Clean algorithm is quite new, we provide entensive information on its use.

In the Multi-scale Clean, the image is cleaned simultaneously with several different beams given by the point spread function convolved with components of various shapes and sizes. The components we use in this implementation are upside-down paraboloids multiplied by first order spheroidal functions (ie, the same functions used in gridding Fourier plane data). The paraboloids are truncated at zero, and the multiplication by the spheroidal function results in a smooth shape with minimal power at long baselines. This shape is scaled to the component sizes specified in setscales. As these functions have finite extent (unlike a Gaussian), they can easily be used with mask images.

For each iteration, the scale size which is able to subtract the most flux is chosen (but with a caveat, see below). The model is then built up out of the spheroidal functions of the various scale sizes. The scale sizes are set by the setscales function, which will permit the user to specify the scale sizes explicitly, or will optionally take the number of scale sizes to clean for and calculate the scale sizes themselves from a power law.

Most images deconvolved with Multi-scale Clean will be dominated by extended structure, and the largest scale size will initially remove the most flux from the dirty image. As the algorithm reduces the residuals on the largest scale, the residuals on the smaller scales will also be reduced, even without cleaning on those size scales (a falling sea sinks all boats). However, at some point, the residual image will be dominated by features on smaller size scales. These smaller features will be both positive and negative (ie, to correct for the largest size scale being the wrong shape for the true emission features). Later in the algorithm, the magnitude of the residuals on all scales will be approximately equal. At this stage, most of the deconvolvable flux has been assimilated into the largest scale size components and detailed corrections to the large scale components must be made. At this point, the user may consider switching to a faster algorithm such as the Clark Clean.

Masking is fully available with Multi-scale Clean. No component is permitted to place any of its wings outside of the user-supplied mask. If the masking were based upon the different scale components' center positions, then the large scale components could place their wings outside the mask, but the smaller scale components would not be able to make fine scale corrections. Hence, the Multi-scale Clean uses a different mask for each different size scale internally. If the mask is too restrictive or the scales are too large, the algorithm may not be able to fit the large scales into the mask at all, and the user is warned of this condition.

Traditional Clean algorithms use a small loop gain such as 0.1 to avoid confusing emission and sidelobes when extended emission is present. However, as MultiScale Clean can image large extended structure in a single spheroidal component, a loop gain in the range 0.5 to 1.0 can be used. If the largest residual oscillates between positive and negative with iteration number, as it can for some brightness distributions which include point sources, a lower loop gain will improve the imaging and the convergence.

A mild bias favoring cleaning small scale emission has been built into the Multi-scale algorithm. To illustrate the requirement of this bias, consider the case of a bright point source with very faint extended emission. Each scale may find its optimal component to subtract at the position of the bright point source, but each successively larger component will integrate more extended flux. Hence, the largeest scale component will be removed from the residuals. If most of the flux were in the point source, then several smaller negative components must be subtracted from the largest component, and finally the point component itself may be removed after the extended emission has been taken care of. To prevent this situation from occurring, we bias the selection of small-sized components.

Note that for all of these functions except fullmsclean, only a quarter of the image may be cleaned. If no mask is set, then the cleaned region defaults to the inner quarter. If a mask larger than a quarter of the image is set, then only the quarter starting at the bottom left corner is used. Algorithm fullmsclean will deconvolve the entire field. This is useful when performing a limited accuracy deconvolution (as needed for example in wide-field imaging) but will diverge if pushed too deep.
The clean threshold may be either absolute ('0.5Jy') or relative ('1%').

Arguments

algorithm	Algorithm to use Allowed:String:'hogbom'—'msclean'—'fullmsclean' Default: 'msclean'
niter	Number of Iterations, set to zero for no CLEANing Allowed:Int Default: 1000
gain	Loop Gain for CLEANing, try 0.7 for msclean or fullmsclean Allowed:Double Default: 0.1
threshold	Flux level at which to stop CLEANing Allowed:String Default: '0Jy'
model	Name of images Allowed:String
mask	Name of mask images used for CLEANing Allowed:String
async	Run asynchronously in the background? Allowed:Bool Default: !dowait

Returns
Bool

Example

```
deco.clean(image='3C273XC1.clean.image', model='3C273XC1.clean.model',
mask='3C283XC1.mask', niter=1000, gain=0.25, threshold=0.03)
```

---

deconvolver.setscales - Function

Package	**general**
Module	**deconvolver**
Tool	**deconvolver**

Set the scale sizes for MultiScale Clean

Synopsis
setscales(scalemethod, nscales, uservector)

Description
Set the scale sizes, all required PSF's and Dirty Images for MultiScale Clean will be calculated. You can either give the number of scales, in which case the the scale sizes are set via a power law, or give a vector of scale sizes in pixels.

Arguments

scalemethod	Method by which scales are set Allowed:String: 'nscales'—'uservector' Default: nscales
nscales	Number of scales Allowed:Int Default: 5
uservector	Vector of scale sizes to use Allowed:Vector of Doubles Default: [0, 3, 10]

Returns
Bool

Example

```
deco.setscales(6);
```

---

deconvolver.ft - Function

Package	**general**
Module	**deconvolver**
Tool	**deconvolver**

Fourier transform the specified model

Synopsis
ft(model, transform, async)

Description
Fourier transform the specified model to an image.

Arguments

model	Name of image Allowed:strings
transform	Name of transform image Allowed:strings
async	Run asynchronously in the background? Allowed:Bool Default: !dowait

Returns
Bool

Example

```
deco.ft(model='3C273XC1.nnls.model', transform='3C273XC1.nnls.model.ft')
```

Fourier transforms the model in the image 3C273XC1.nnls.model

---

deconvolver.restore - Function

Package	**general**
Module	**deconvolver**
Tool	**deconvolver**

Restore the residuals

Synopsis
restore(model, image, bmaj, bmin, bpa, async)

Description

Restore the residuals to a smoothed version of the model. The model images are convolved with the specified Gaussian beam and then the residual images are added. If the beam is not supplied, one will be fit to the PSF.

Arguments

model	Name of input model Allowed:String
image	Name of output restored image Allowed:String
bmaj	Major axis of beam Allowed:Quantity record
bmin	Minor axis of beam Allowed:Quantity record
bpa	Position angle of beam Allowed:Quantity record
async	Run asynchronously in the background Allowed:Bool Default: !dowait

Returns

Bool

Example

```
deco.restore(model='3C273XC1.clean', image='3C273XC1.clean.restored',
bmaj='2.0arcsec', bmin='2.0arcsec')
```

---

deconvolver.residual - Function

Package	**general**
Module	**deconvolver**
Tool	**deconvolver**

Find the residuals

Synopsis

residual(model, image, async)

Description
Calculate the residuals corresponding to the model. componentlist.

Arguments

model	Names of input models Allowed:String
image	Names of output residual images Allowed:String
async	Run asynchronously in the background Allowed:Bool Default: !dowait

Returns
Bool

Example

```
deco.residual(model='3C273XC1.clean', complist='3C273XC1.cl',
image='3C273XC1.clean.residual')
```

---

deconvolver.smooth - Function

Package	**general**
Module	**deconvolver**
Tool	**deconvolver**

smooth the image

Synopsis
smooth(model, image, bmaj, bmin, bpa, normalize, async)

Description
The model image is convolved with the specified Gaussian beam. By default (normalize=T), the beam volume is normalized to unity so that the smoothing is flux preserving. The smoothing used in restoration is not normalized.

Arguments

model	Name of input model Allowed:String
image	Name of output smoothed image Allowed:String
bmaj	Major axis of beam Allowed:Quantity record
bmin	Minor axis of beam Allowed:Quantity record
bpa	Position angle of beam Allowed:Quantity record
normalize	Normalize volume of psf to unity Allowed:Bool Default: T
async	Run asynchronously in the background Allowed:Bool Default: !dowait

Returns
Bool

Example

```
- deco.smooth(model='3C273XC1.clean', image='3C273XC1.clean.restored',
bmaj='2.0arcsec', bmin='2.0arcsec')
```

---

deconvolver.mem - Function

Package	**general**
Module	**deconvolver**
Tool	**deconvolver**

Make the mem image

Synopsis
mem(entropy, niter, sigma, targetflux, constrainflux, model, prior, mask, imageplane, async)

Description
Makes a mem image using the Cornwell-Evans algroithm, using either

maximum entropy (entropy) or maxmimum emptiness (emptiness). The maximum entropy algorithm is the default. You can restart a MEM deconvolution on an existing model image, but the alpha and beta parameters are not yet saved.

Mask images can be used to restrict where the algorithm puts flux. A prior, or bias, image can provide a priori information to the algorithm and effectively limit the support as well as a mask. The prior image can be constructed by smoothing an existing estimate for the brightness distribution and clipping. Any pixel values below 1e-6 will be clipped to this level, so zero or negative pixels will not cause problems.

Currently, only one Stokes parameter may be deconvolved at a time. Stokes I images can be deconvolved with either maximum entropy or maxmimum emptiness. Stokes Q, U, or V should be deconvolved with maxmimum emptiness, which permits negative pixel values. Joint polarization MEM deconvolution is planned for the future.

The mem entropies possible are:

entropy The smoothness of the image, relative to some prior (also called default or bias) image is maximized. The functional form of the entropy is $H = \sum Iln(I/M)$, where I is the mem image brightness and M is the prior image. As the prior image is positive definite, the entropy constrains the mem image pixels to be positive, hence only stokes I can be imaged.

emptiness The number of pixels with absolute value of the flux greater than the noise level is minimized. This treats positive and negative pixel values equally, so it is appropriate for any Stokes image.

This MEM algorithm works in the image plane (ie, is ignorant of visibility data), but performs the convolution by multiplication in the Fourier plane. Not to be confused with this usage of the term "image plane", some problems are "image plane" problems, such as a single dish performing On-The-Fly mapping. Independent noise is added at each integration as the beam sweeps over the object (ie, in the image plane). This can lead to a noise signal at non-physically large spatial frequencies. This non-physical signal can be removed by convolving the residual image with the PSF. Also key to this problem is that the PSF is of finite extent, permitting the deconvolution of nearly the entire dirty image rather than just the inner quarter. These options are accessed by setting `imageplane` to T.

Arguments

entropy	entropy to use Allowed:String:'entropy'—'emptiness' Default: 'entropy'
niter	Number of Iterations, set to zero for no MEMing Allowed:Int Default: 20
sigma	Noise level to try to achieve Allowed:String Default: '0.001Jy'
targetflux	Total image flux to try to achieve Allowed:String Default: '1.0Jy'
constrainflux	Use targetflux as a constraint? (or starting flux) Allowed:Bool Default: F
model	Name of input/output model image Allowed:String
prior	Name of prior (default) image used for mem Allowed:String
mask	Mask image restricting emission (all pixels 0 or 1) Allowed:String
imageplane	Is this an image plane problem (like single dish)? Allowed:Bool Default: F
async	Run asynchronously in the background? Allowed:Bool Default: !dowait

Returns
Bool

Example

```
deco.mem(entropy='entropy', niter=30, sigma=0.01, targetflux=10.0,
model='3C273XC1.mem.image', prior='3C283XC1.prior')
```

---

deconvolver.makeprior - Function

Package	**general**
Module	**deconvolver**
Tool	**deconvolver**

Make the mem's prior image, or make a mask

Synopsis
makeprior(prior, templateimage, clipfrom, clipto, blc, trc, async)

Description
Makes a prior image for the mem function. A general way to make a prior image is to start with a low resolution image, obtained from a smaller array configuration or a lower frequency observation, from another image which has been smoothed, or from a single dish image. The low resolution image can then be doctored via clipping and regioning to make it acceptable for the mem function.
Currently, only one Stokes parameter may be used at a time.

Arguments

prior	output prior image Allowed:String
templateimage	starting point for prior image Allowed:String
clipfrom	Clip any pixel below this level Allowed:String Default: '0.0Jy'
clipto	Any clipped pixel will be given this value Allowed:String Default: '0.0Jy'
blc	Bottom left hand corner for box; outside box is clipped Allowed:integer[]
trc	Top right hand corner for box; outside box is clipped Allowed:integer[]
async	Run asynchronously in the background? Allowed:Bool Default: !dowait

Returns
Bool

Example

```
deco.makeprior(prior='3C283XC1.prior', templateimage='3C283XC1.mem.smooth',
clipfrom='0.01Jy', clipto='0.0001Jy', blc=[100,100], trc=[150,150])
```

1.2 images - Module

Access and analysis of images
include "images.g"

Description
This module contains functionality to access, create, and analyze AIPS++ images. It offers both basic services and higher-level packaged tools.
The available tools in this module are

- Image - create, manipulate and analyze images (no default tool). This `tool` provides a range of low to medium level services.
- Regionmanager - create and manipulate `regions-of-interest`(default tool is **drm**).
- Coordsys - create and manipulate Coordinate Systems (no default tool).
- Imagefitter - interactively fit 2-D models to an image (no default tool).
- Imageprofilefitter - interactively fit models to 1-D profiles from an image (no default tool).
- Imagepol - this offers specialized polarimetric analysis of images.
- Imageprofilesupport - this offers plotting of 1-D profiles with built in abcissa unit conversions. This is not a high-level user `tool`. It is a low-level `tool` used in high-level `tools` such as Imageprofilefitter.

Images
We refer to an AIPS++ `image file` when we are referring to the actual data stored on disk. The name that you give an AIPS++ `image file` is actually the name of a directory containing a collection of AIPS++ tables which together constitute the `image file`. But you only need to refer to the directory name and you can think of it as one *logical* file. Some images don't have an associated disk file. These are called "virtual" images (e.g. an image that is temporary and entirely in memory). Whenever we use the word "image", we are just using it in a generic sense.
Images are manipulated with an Image tool.

Pixel mask
A `pixel mask` specifies which pixels are to be considered good (mask value `T`) or bad (mask value `F`). For example, you may have imported a FITS file which has blanked pixels in it. These will be converted into `pixel mask` elements whose values are bad (`F`). Or you may have made an error analysis of an image and computed via a statistical test that certain pixels should be masked out for future analysis.
An `image file` may contain zero, one, or more `pixel masks`. However, only one mask will be designated as the default mask and be applied to the data.

For more details, see the Image tool.

Region-of-interest
A `region-of-interest` or simply, region, designates which pixels of the image you are interested in for some (generally) astrophysical reason. This complements the `pixel mask` which specifies which pixels are good or bad (for statistical reasons). `Regions-of-interest` are generated and manipulated with the Regionmanager tool.
Briefly, a `region-of-interest` may be either a simple shape such as a multi-dimensional box, or a 2-D polygon, or some compound combination of `regions-of-interest`. For example, a 2-D polygon defined in the X and Y axes extended along the Z axis, or perhaps a union or intersection of regions. See the Regionmanager documentation for more details on regions.

Coordinates
We will often refer to (absolute) pixel coordinates. Consider a 2-D image with shape [10,20]. Then our model is that the centre of the bottom-left corner pixel has pixel coordinate [X,Y] = [1,1]. The centre of the top-right corner pixel has pixel coordinate [X,Y] = [10,20].
When a physical Coordinate System (e.g. an RA/DEC direction coordinate) is attached to an image, then we can convert pixel coordinates to a world (or physical) coordinate.
The Coordsys `tool` is available for manipulating Coordinate Systems. You can recover the Coordinate System from an image into a Coordsys `tool` via the Image function coordsys.
For more details, see the Image and Coordsys tools.

Lattice Expression Language (LEL)
This allows you to manipulate expressions involving images. For example, add this image to that image, or multiply the miniumum value of that image by the square root of this image. The LEL syntax is quite rich and is described in detail in note 223.
LEL is used in several of the Image tool functions.

Example

Here is a simple example to give you the flavour of using the `image` module. Suppose we have an image FITS disk file; we would like to convert it to an AIPS++ `image file` (stores the image in an AIPS++ table), look at the header information and store it in a *Glish* record for future use, work out some statistics and display the image with the Viewer.

```
include 'images.g'                                                  # 1
im:=imagefromfits(outfile='data/ne.xyv', infile='$FITS/ne.FITS')    # 2
im.summary()                                                        # 3
im.statistics()                                                     # 4
```

```
im.view()                                                            # 5
im.done()                                                            # 6
```

1. This gives us access to the Image tool code.
2. Convert (using the `imagefromfits` constructor) an image FITS disk file called *'ne.FITS'* in the directory specified by the environment variable *'$FITS'* to an AIPS++ `image file` called *'ne.xyv'* in the subdirectory called *data*. The AIPS++ `image file` is now associated with an `image tool` called **im** which was constructed in this process.
3. This summarises (to the Logger) the basic header information in the image (name, masks, regions, brightness units, coordinates etc).
4. This `tool function` evaluates basic statistics from the entire image.
5. This displays the image with the Viewer.
6. This destroys the Image `tool`, but not its associated disk `image file`.

Tools

coordsys	Operations on CoordinateSystems
image	Operations on images
imagefitter	Fit models to images of the sky
imagepol	Polarimetric analysis of images
imageprofilefitter	Fit models to 1D profiles from images
imageprofilesupport	Plot profiles with axis label conversions
regionmanager	Create and manipulate regions of interest

Functions

coordsystest	Run test suite for Coordsys tool
coordsystools	Find the list of current global Coordsys tools
imagedemo	Demonstrate the use of an image
imagedones	Done all global or specified Image tools
imagefiles	Find the list of image files in the given directory
imagefittertest	Run test suite for Imagefitter tool
imagepoltest	Run test suite for Imagepol tool
imageprofilesupporttest	Run test suite
imagetest	Run test suite for Image tool
imagetools	Find the list of current global image tools
is_coordsys	Is this variable a Coordsys tool ?
is_image	Is this variable an image tool ?
is_imagepol	Is this variable an imagepol tool ?
is_region	Is this a valid region tool?

%