Introduction to Imaging in CASA

Mark Lacy
(NRAO)

Atacama Large Millimeter/submillimeter Array
Expanded Very Large Array
Robert C. Byrd Green Bank Telescope
Very Long Baseline Array

Overview

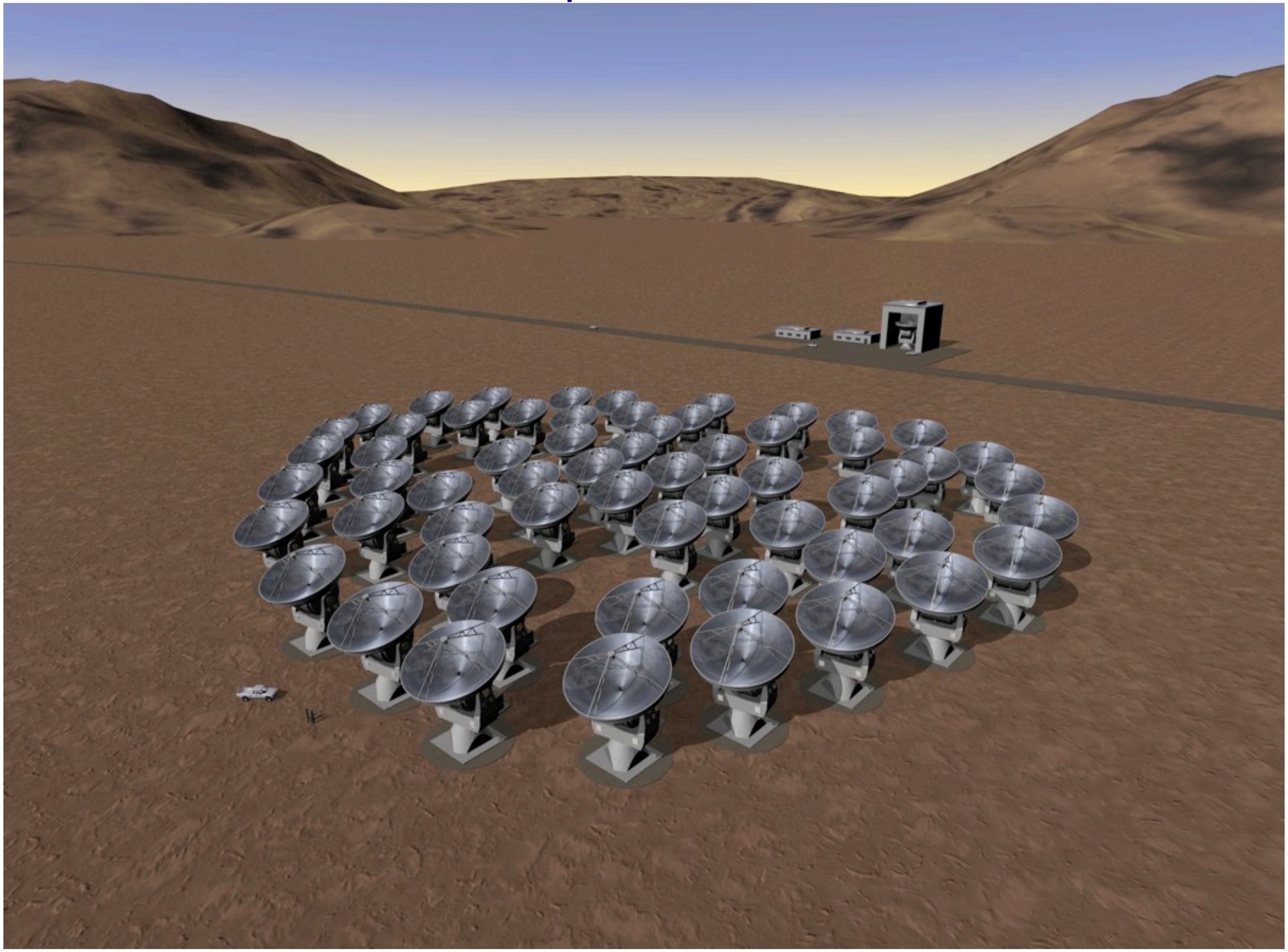
- Goals of this talk:
  - Gain some intuition for interferometric imaging
  - Introduce deconvolution in CASA (CLEAN)
  - Introduce various imaging methods available in CASA

- More formal description of imaging available in NRAO Synthesis Imaging Workshop lectures

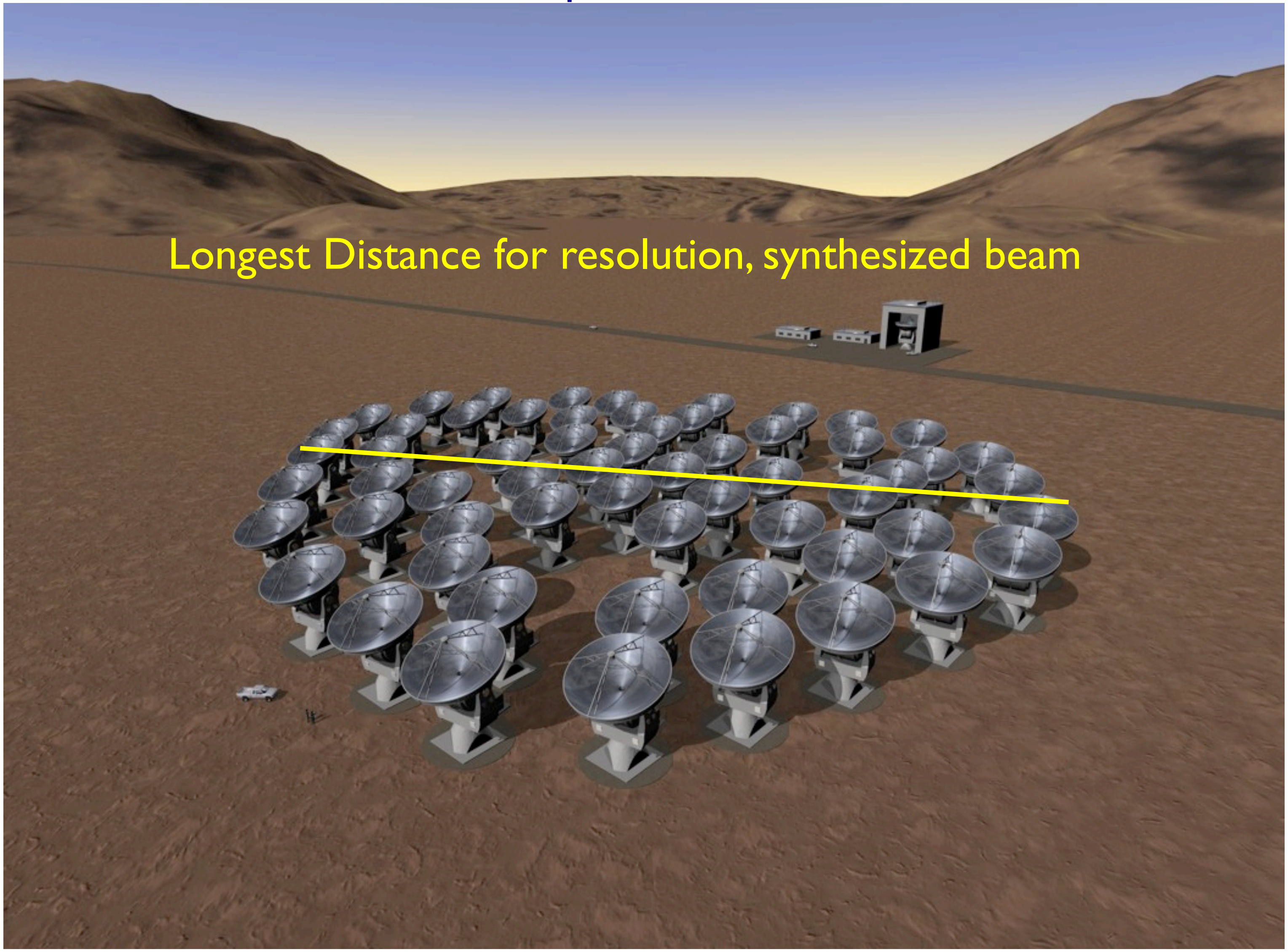
NRAO

Single dish: diameter is responsible for sensitivity, field of view, resolution

Interferometer: takes this apart

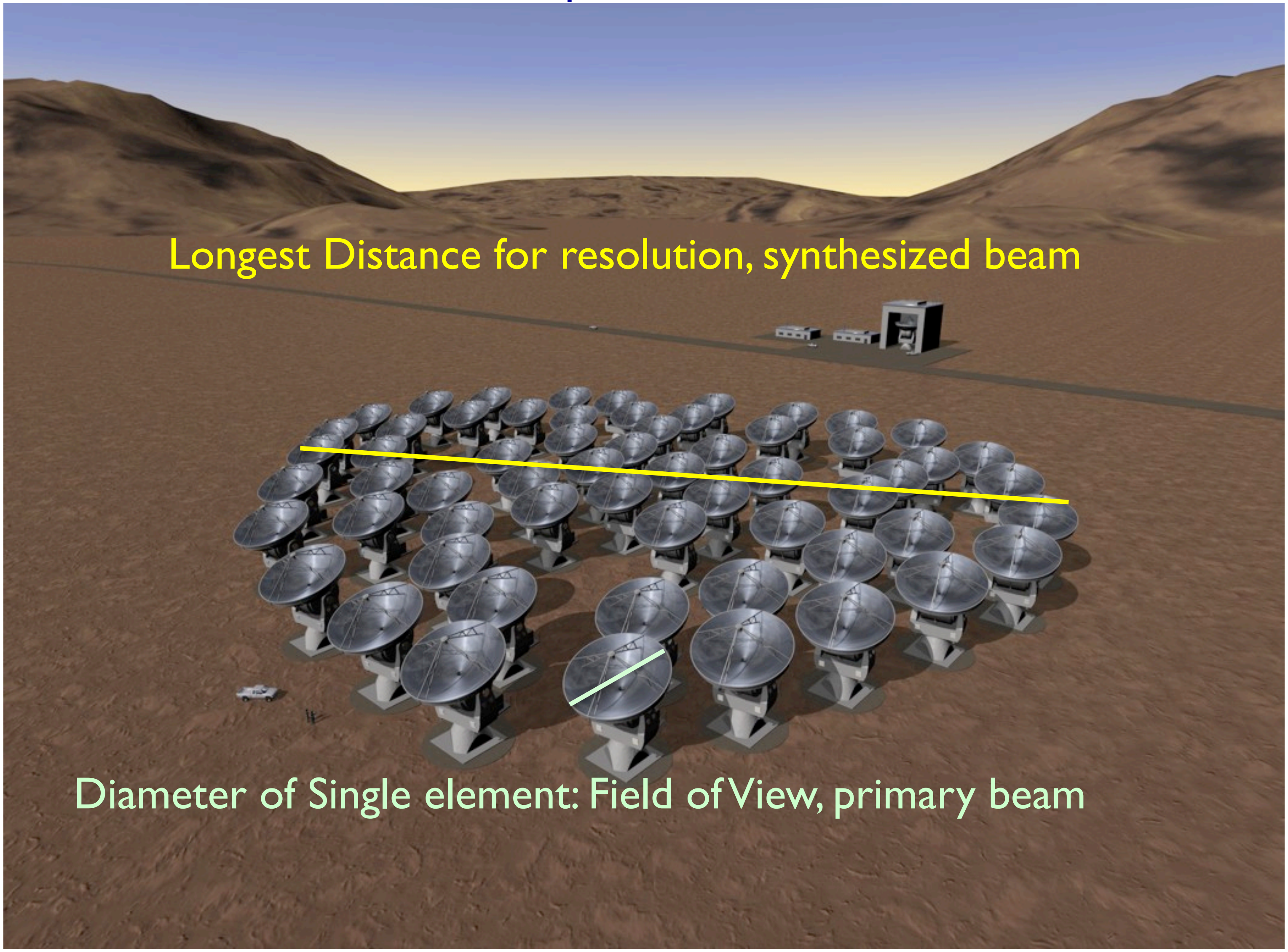


Single dish: diameter is responsible for sensitivity, field of view, resolution
Interferometer: takes this apart



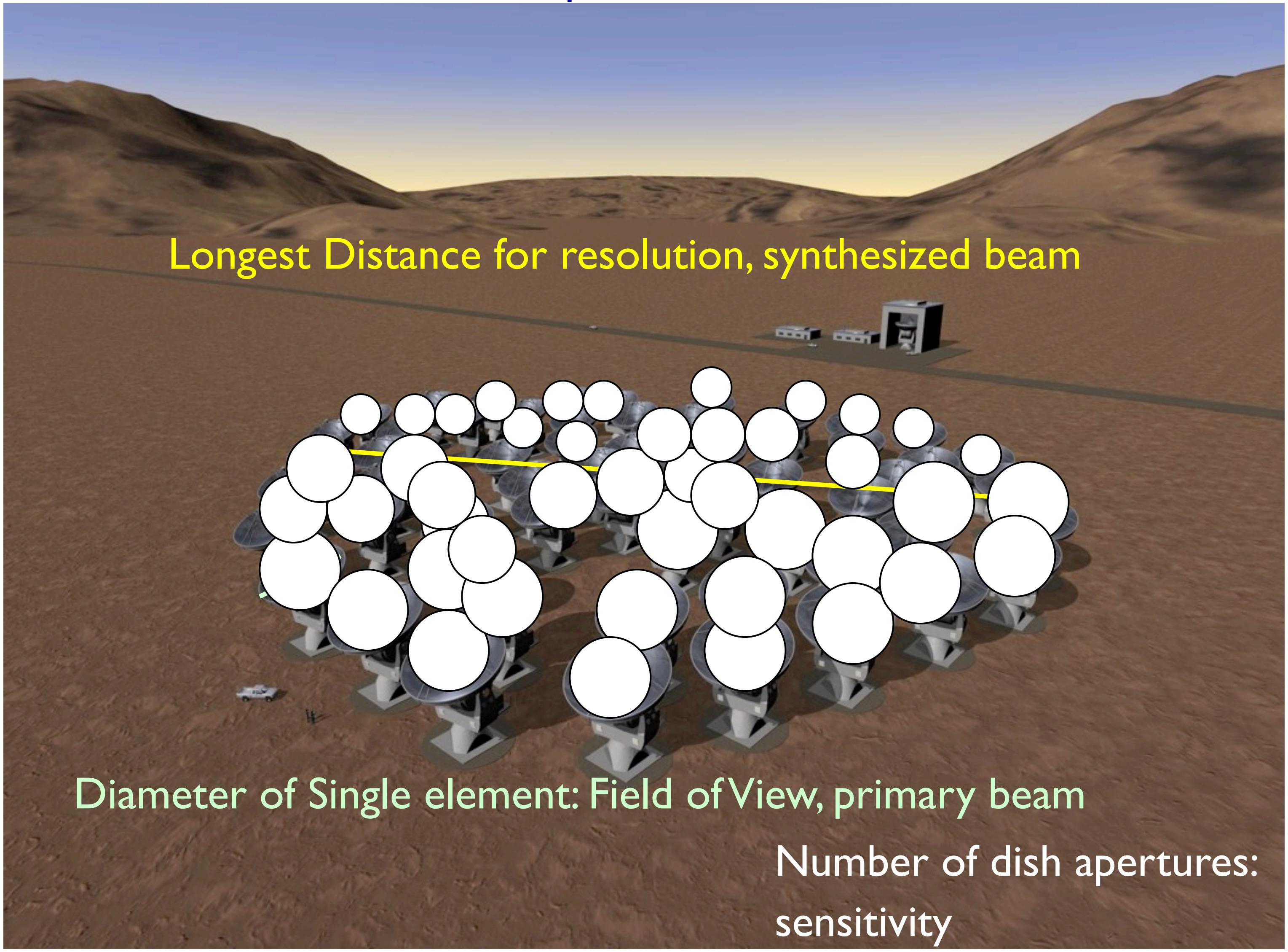
Single dish: diameter is responsible for sensitivity, field of view, resolution

Interferometer: takes this apart



Single dish: diameter is responsible for sensitivity, field of view, resolution

Interferometer: takes this apart

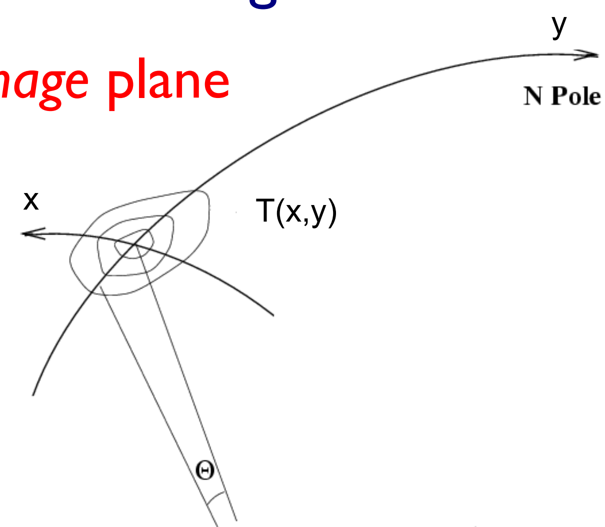


From Sky Brightness to Visibility

1. An interferometer measures the interference pattern observed by pairs of apertures
2. The interference pattern is directly related to the source brightness. In particular, for small fields of view the complex visibility, V(u,v), is the 2D Fourier transform of the brightness on the sky, T(x,y)

   (van Cittert-Zernike theorem)

image plane



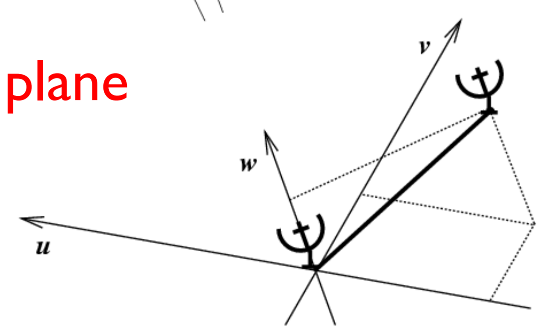
Fourier space/domain

$$V(u,v) = \int\int T(x,y) e^{2\pi i(ux+vy)} dxdy$$

$$T(x,y) = \int\int V(u,v) e^{-2\pi i(ux+vy)} dudv$$

Image space/domain

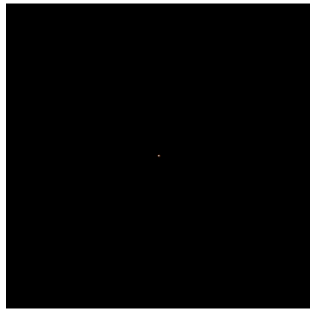
uv plane



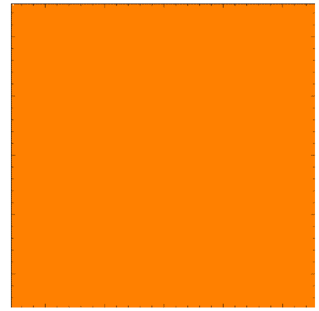
Some 2D Fourier Transform Pairs

T(x,y) | Amp{V(u,v)}

δ Function

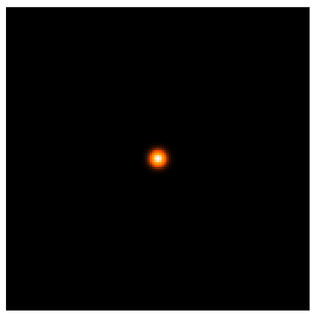


$\rightleftharpoons$



Constant

Gaussian

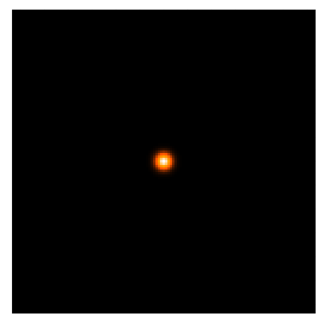


$\rightleftharpoons$

Gaussian

Gaussian

$\rightleftharpoons$



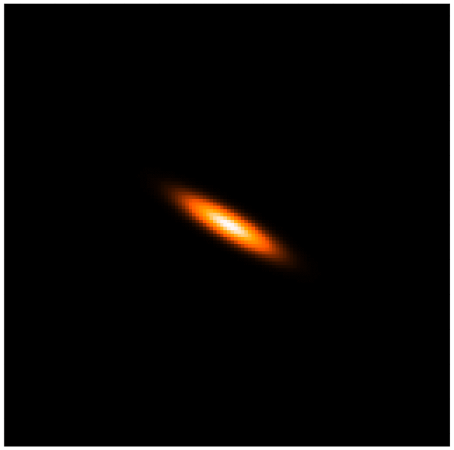
Gaussian

narrow features transform to wide features (and vice-versa)

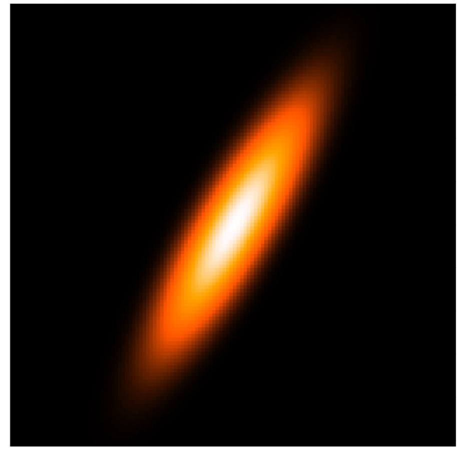
More 2D Fourier Transform Pairs

T(x,y) | Amp{V(u,v)}

elliptical Gaussian



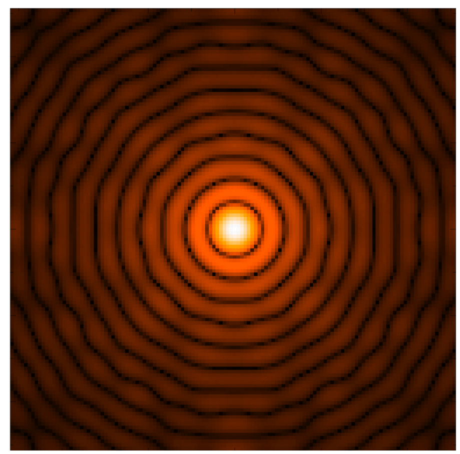
$\rightleftharpoons$



elliptical Gaussian

Disk

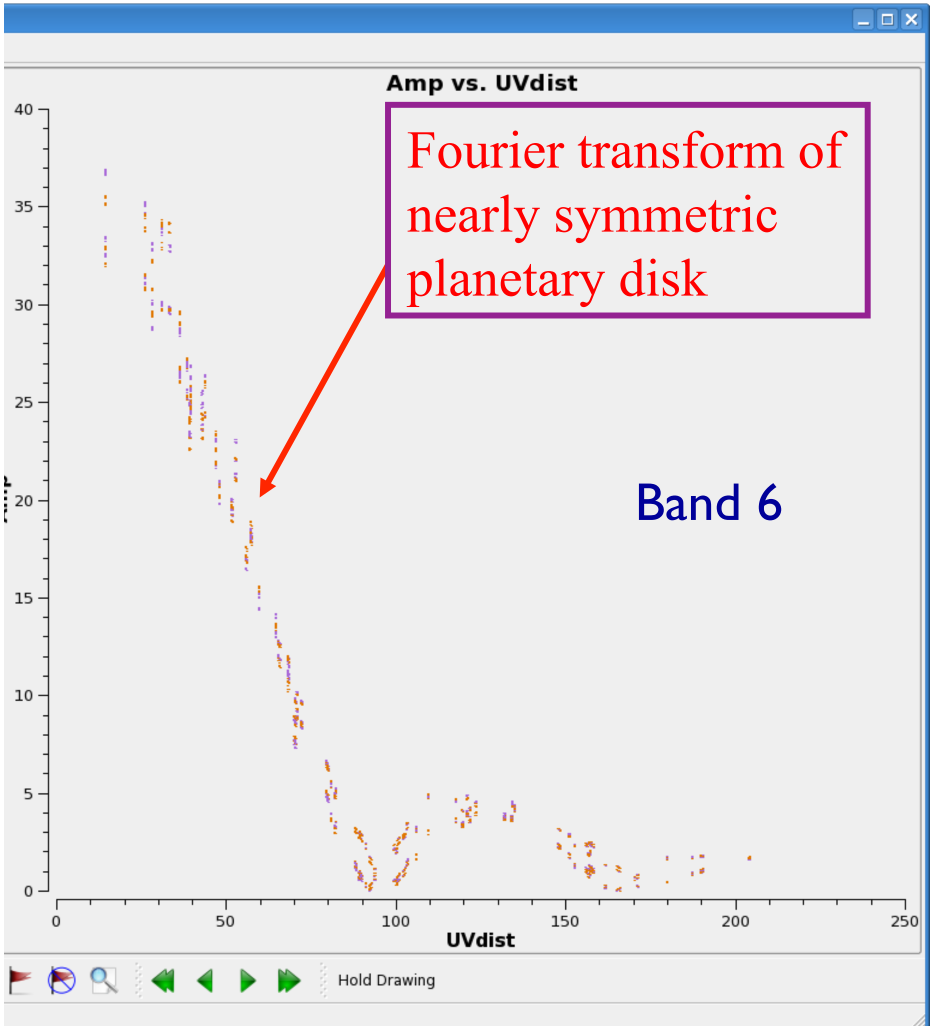
$\rightleftharpoons$

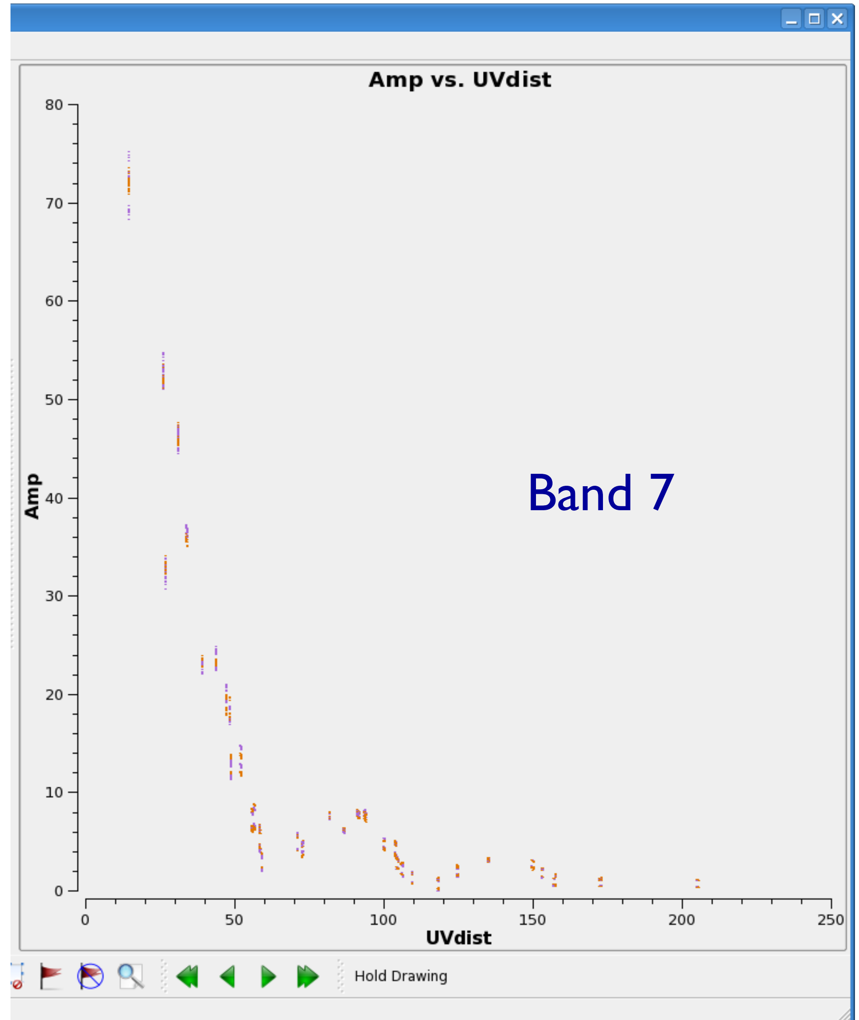


Bessel

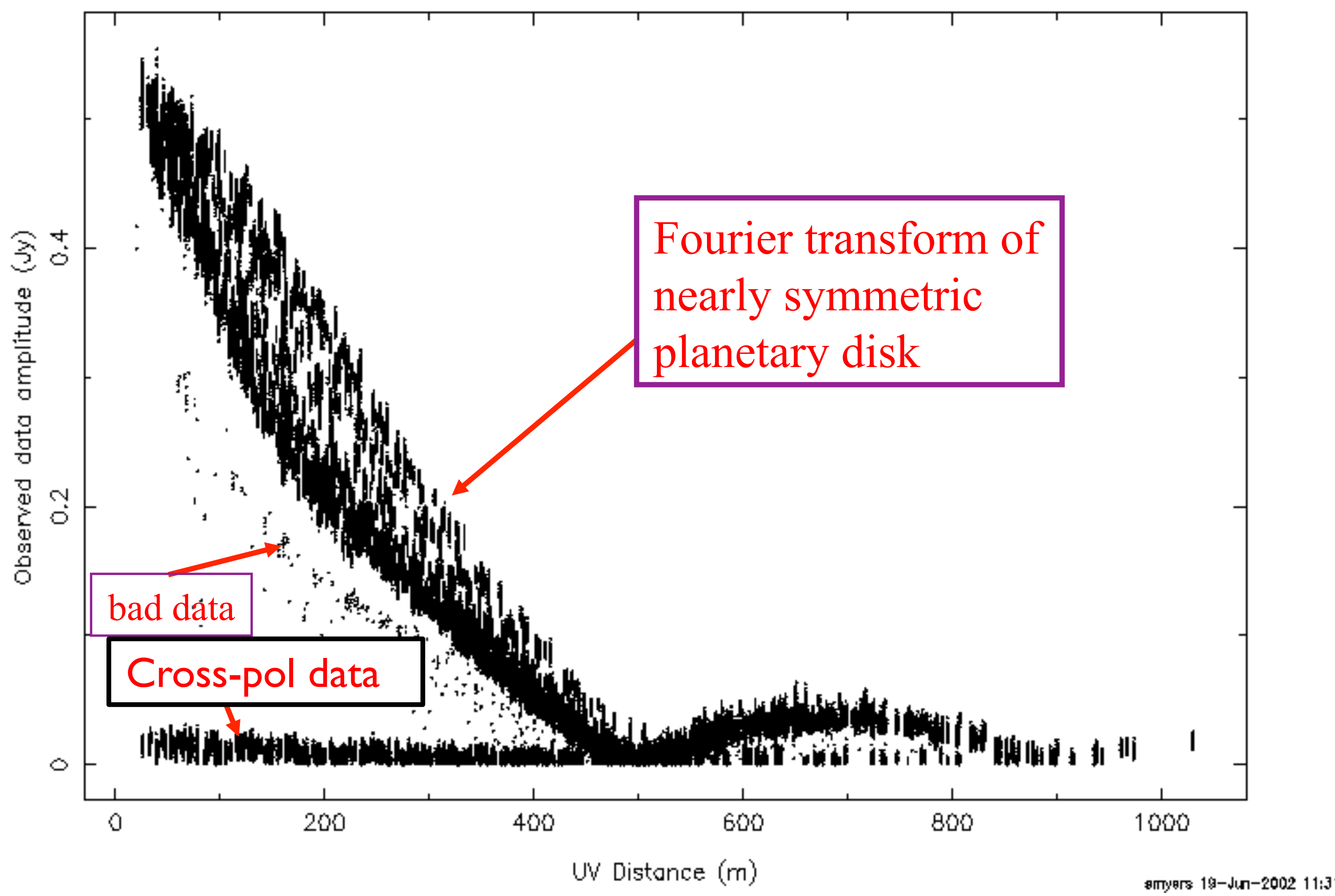
sharp edges result in many high spatial frequencies
(sinc function, “ringing”, Gibbs phenomenon)

ALMA observes planetary disk



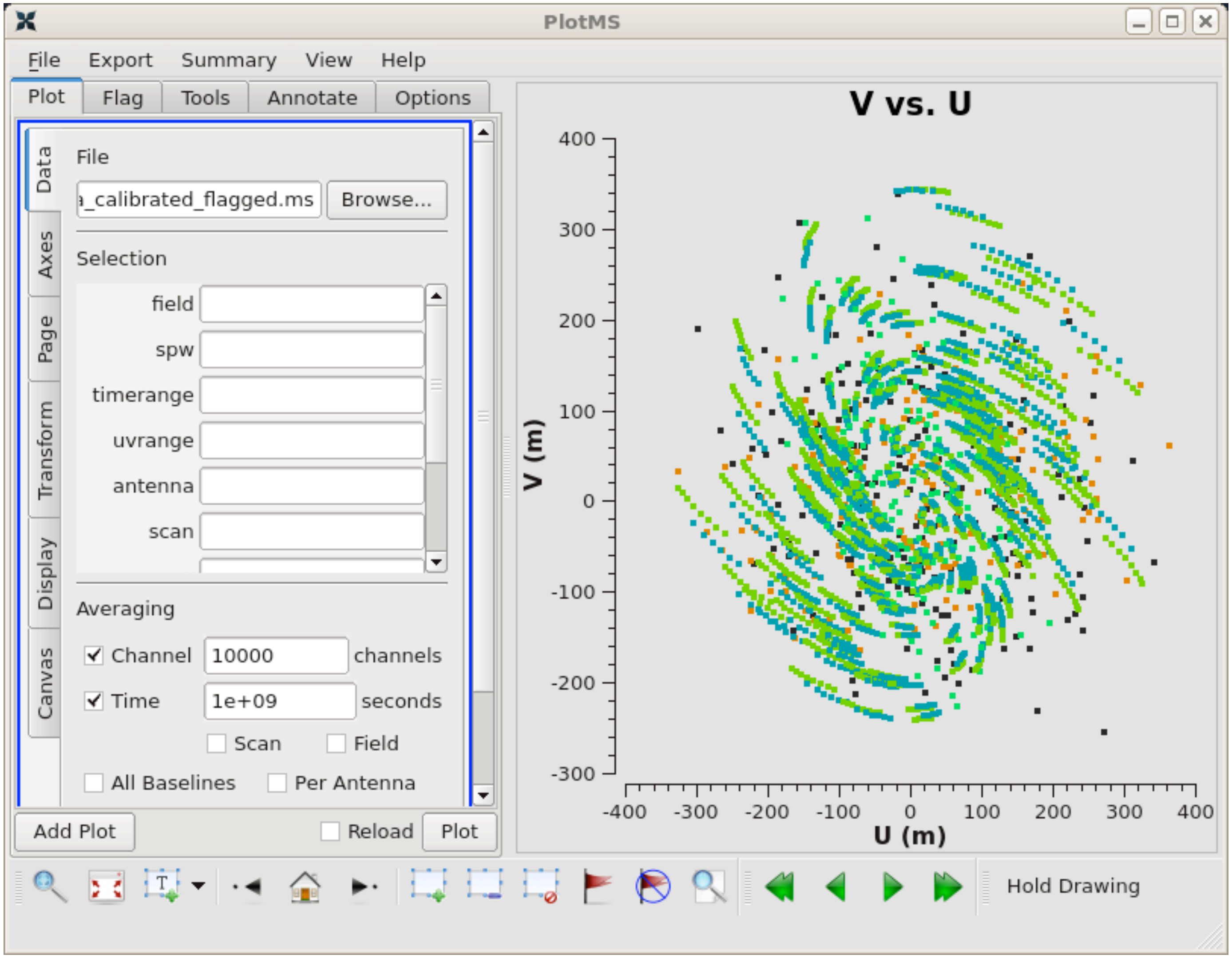


VLA observes Jupiter (6cm)



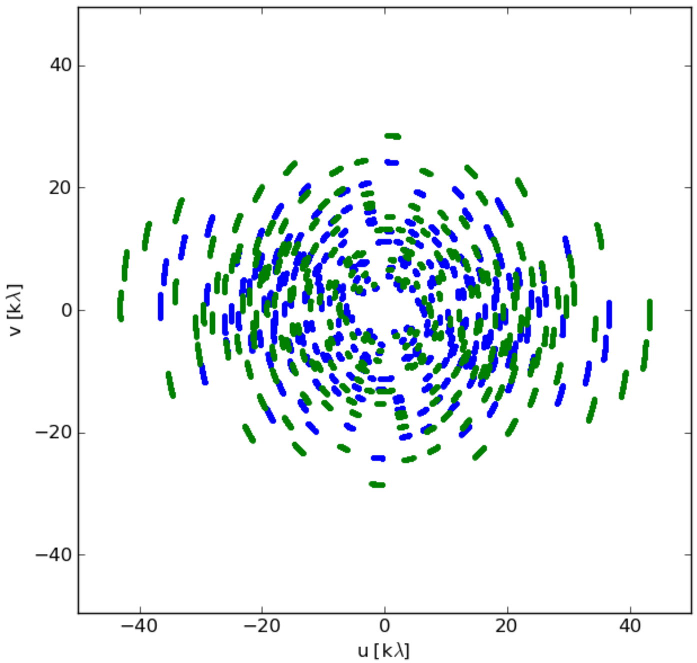
S.T. Myers

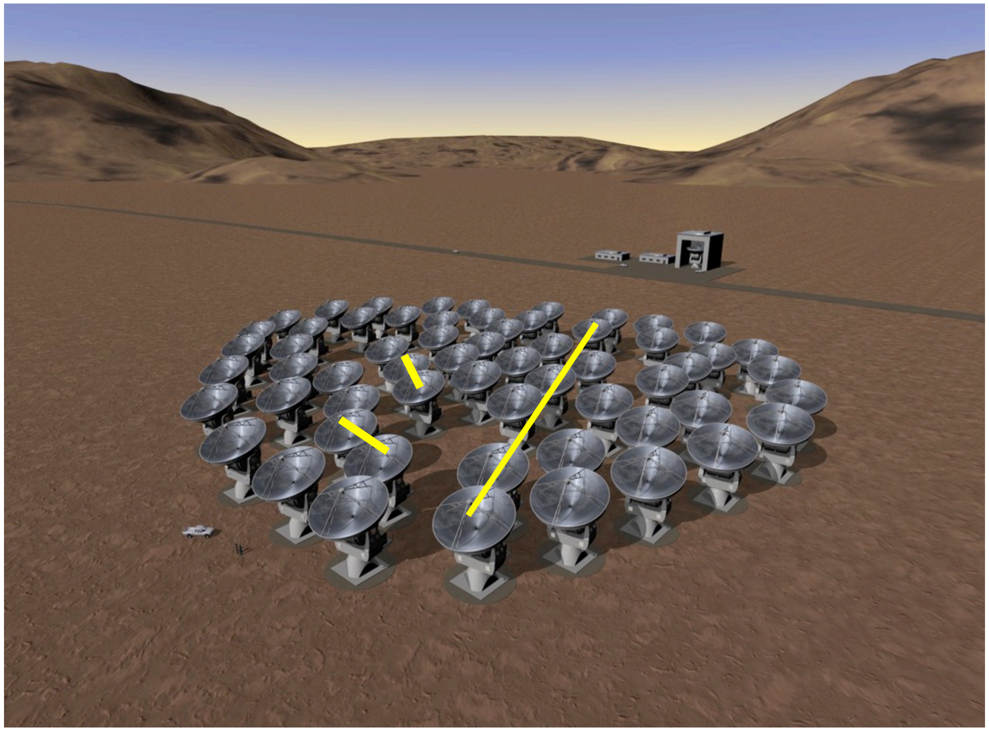
Plotms: Versatile examination of UV data



Sampling Function

Interferometers cannot see the entire Fourier/uv domain. But each antenna pair samples one spot: ➔ **imperfect image**





Small uv-distance: short baselines (measure extended emission)
Long uv-distance: long baselines (measure small scale emission)
Orientation of baseline also determines orientation in the uv-plane
Each visibility has a phase and an amplitude

NRAO

Dirty Images from a Dirty Beam

- We sample the Fourier domain at discrete points

  $$B(u,v) = \sum_k (u_k, v_k)$$

- This sampling function is applied to the source visibilities, and when Fourier Transformed, gives us the "dirty image":

  $$T^D(x,y) = FT^{-1}\{B(u,v) \times V(u,v)\}$$

- The convolution theorem tells us that we can write this as a convolution of the true image with a "dirty beam":

  $$T^D(x,y) = b(x,y) \otimes T(x,y)$$

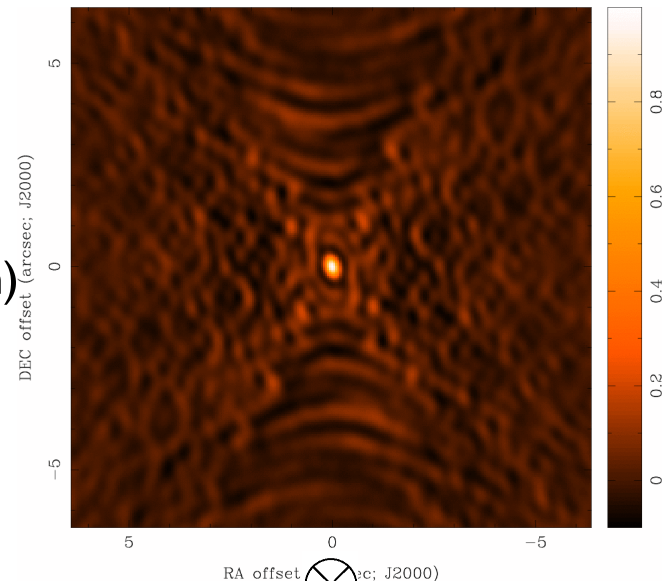
- Where the point spread function ("dirty beam") is

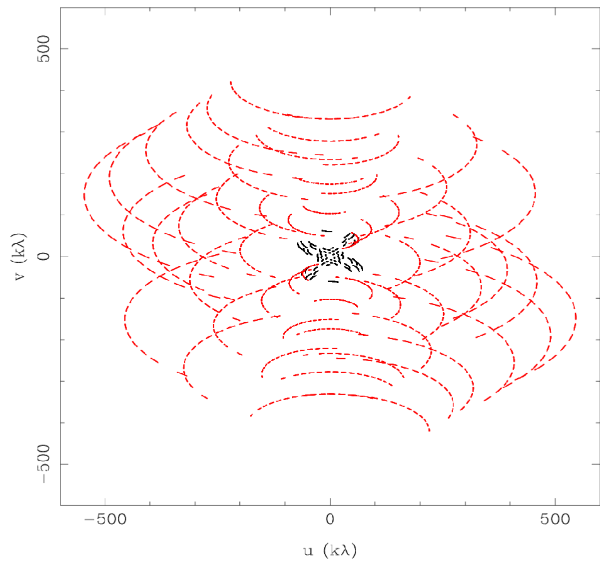
  $$b(x,y) = FT^{-1}\{B(u,v)\}$$

NRAO

Dirty Beam and Dirty Image

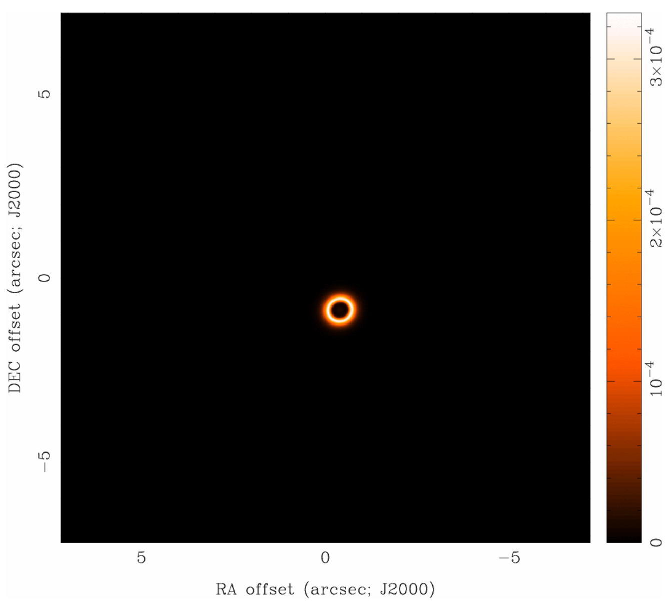
b(x,y)
(dirty beam)

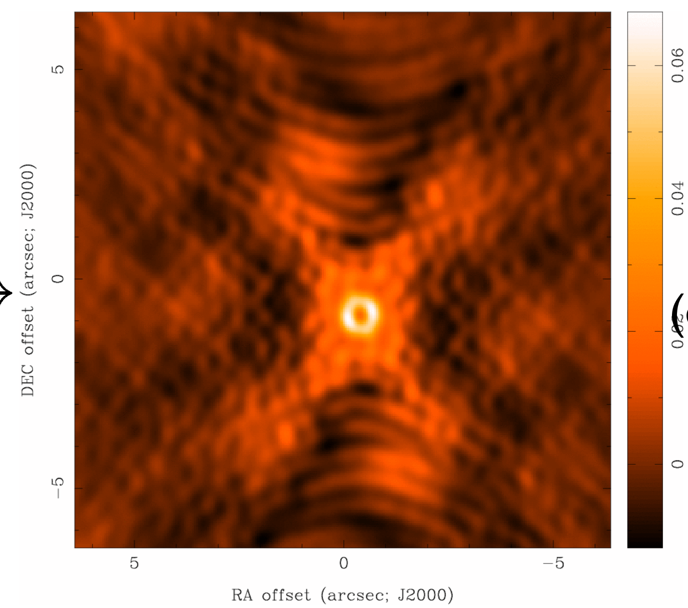




B(u,v)

T(x,y)



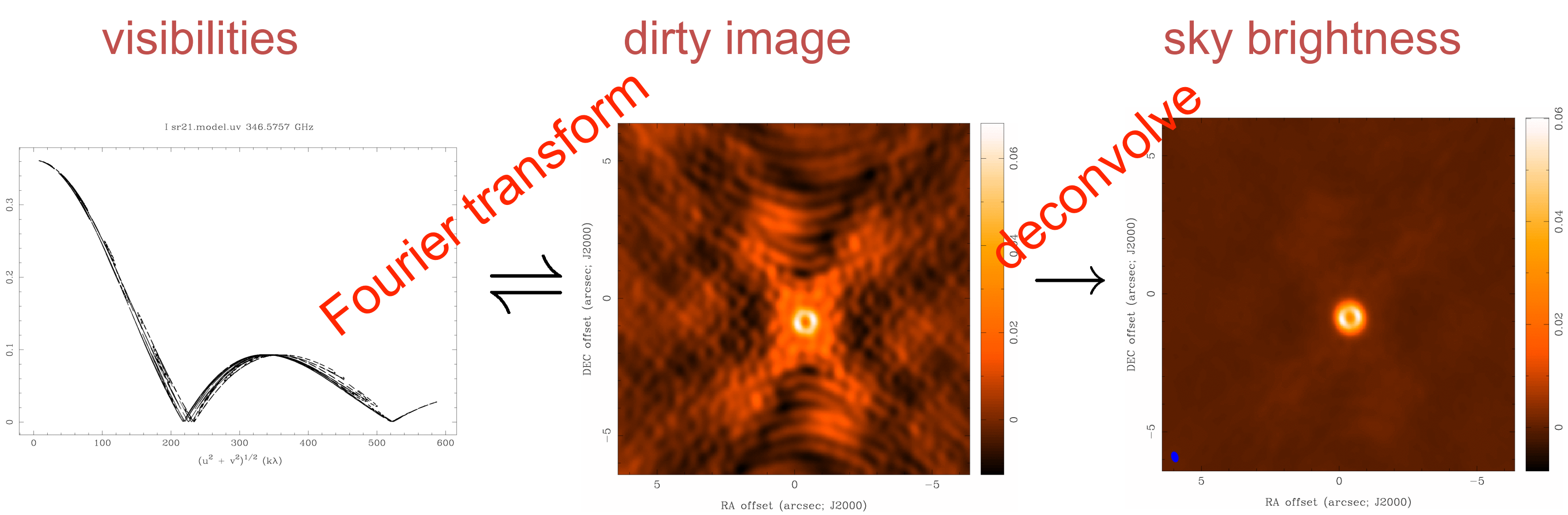


TD(x,y)
(dirty image)

How to analyze (imperfect) interferometer data

Image plane analysis

- dirty image TD(x,y) = Fourier transform { V(u,v) }
- deconvolve b(x,y) from TD(x,y) to determine (model of) T(x,y)

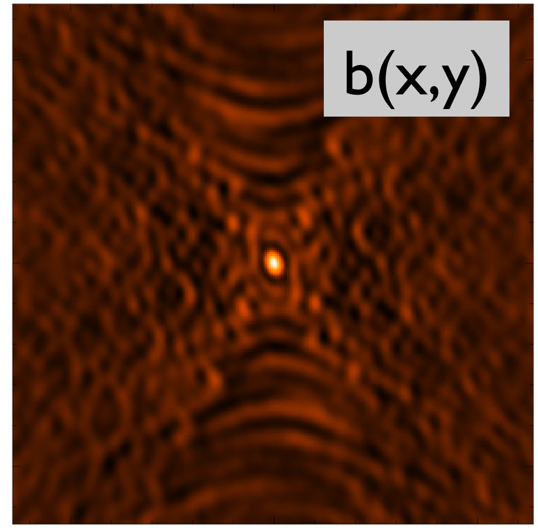


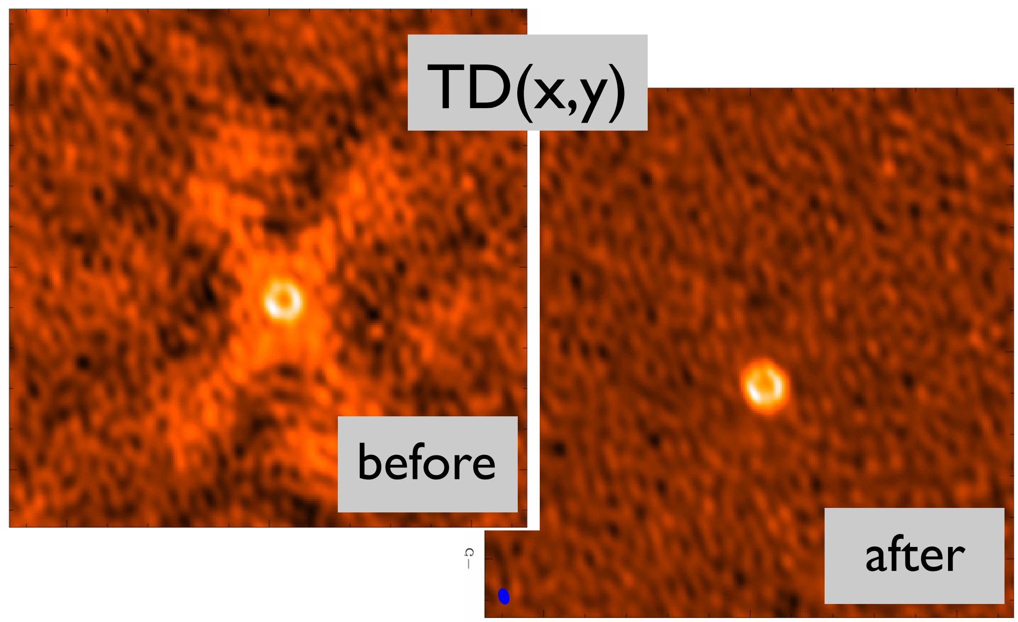
Basic CLEAN Algorithm

A. Initialize a *residual* map to the dirty map

1. Start loop
2. Identify strongest feature in *residual* map as a point source
3. Add this point source to the clean component list
4. Convolve the point source with b(x,y) and subtract a fraction g (the loop gain) of that from *residual* map
5. If stopping criteria not reached, do next iteration

B. Convolve *Clean component* (cc) list by an estimate of the main lobe of the dirty beam (the "Clean beam") and add *residual* map to make the final "restored" image





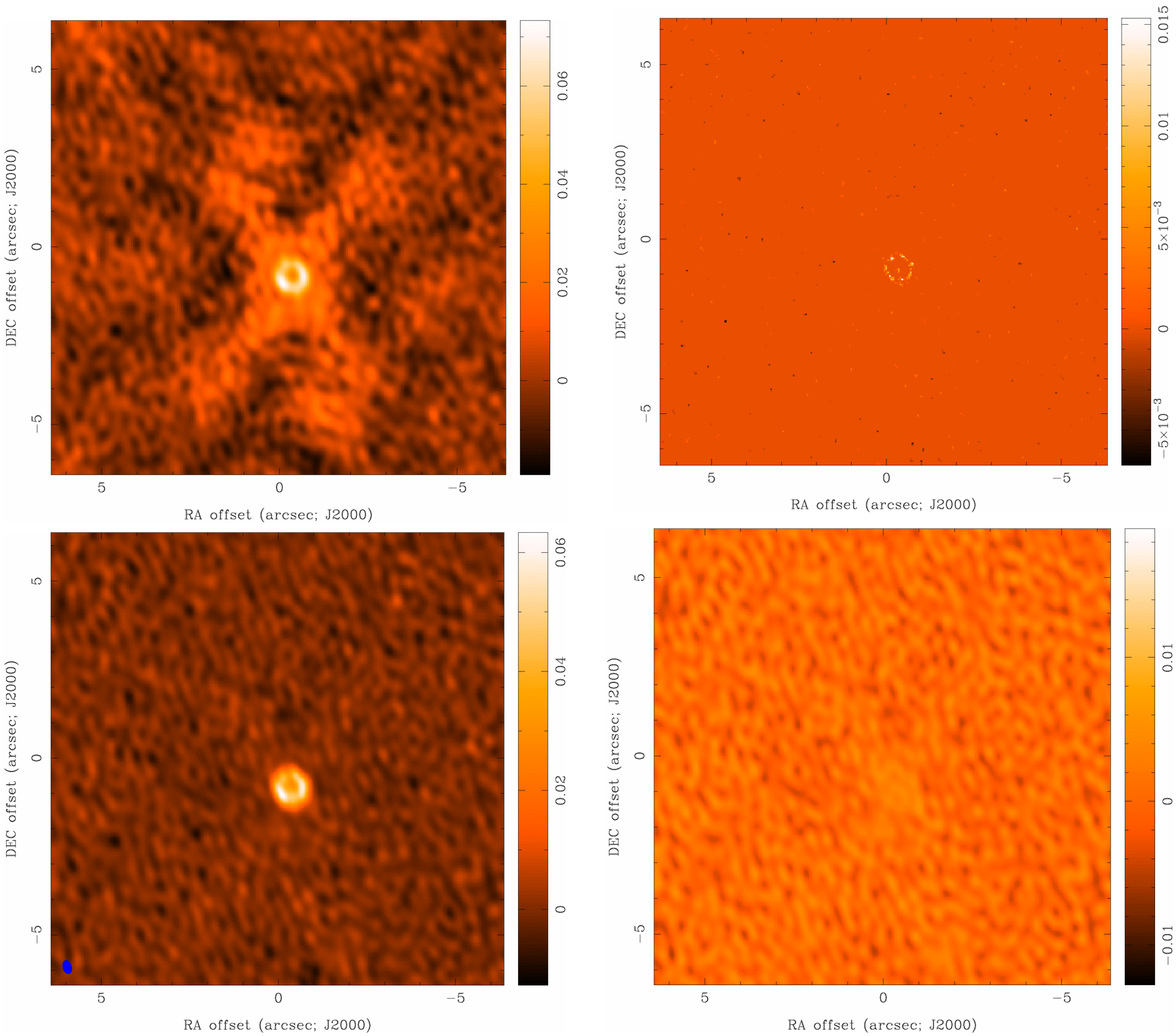
Basic CLEAN Algorithm (cont.)

- Stopping criteria
  - *residual* map max < multiple of rms (when noise limited)
  - *residual* map max < fraction of dirty map max (dynamic range limited)
  - max number of clean components reached (no justification)
- Loop gain
  - good results for g ~ 0.1 to 0.3
  - lower values can work better for smoother emission, g ~ 0.05
- Easy to include *a priori* information about where to search for clean components ("clean boxes")
  - very useful but potentially dangerous!

CLEAN

TD(x,y)

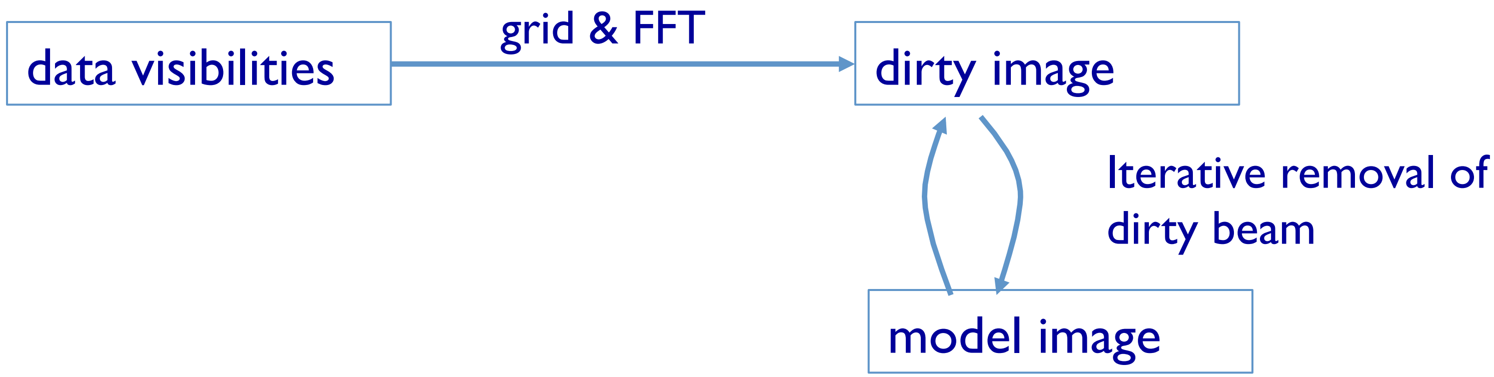
CLEAN model



restored image

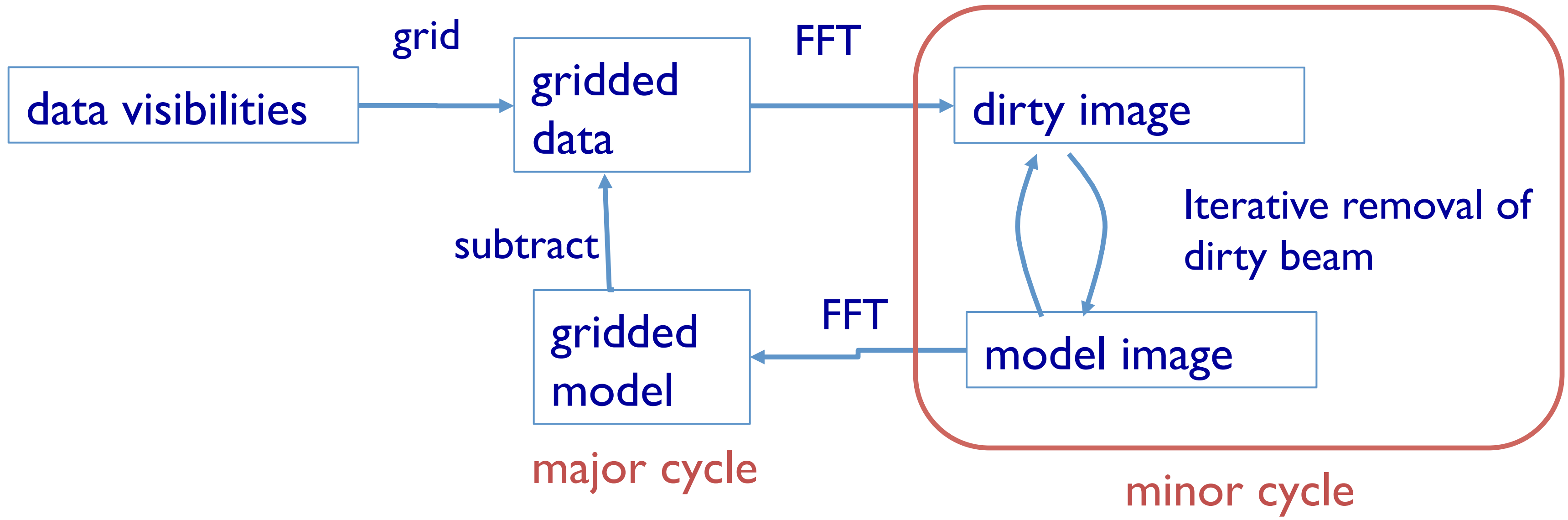
residual map

Deconvolution algorithms : Hogbom



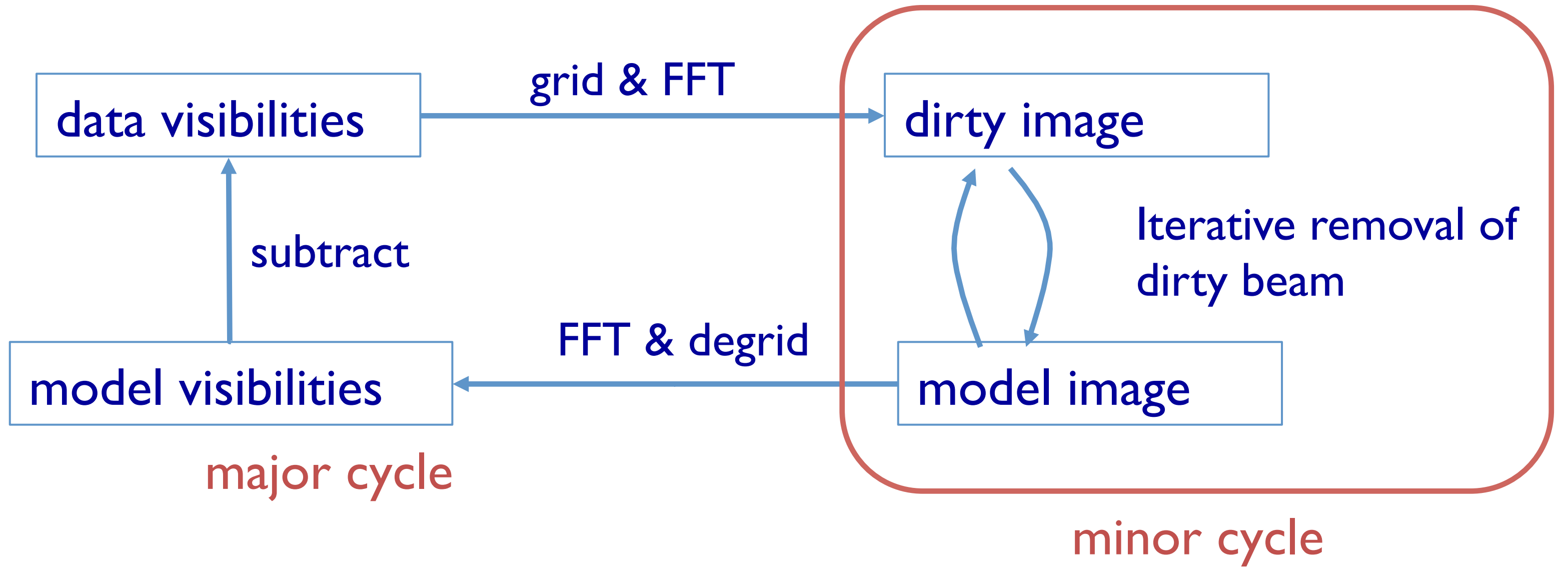
- Subtracts full PSF in image domain
- For complex images, errors can build

Deconvolution algorithms : Clark



- Subtracts truncated PSF in image domain
- Periodically subtracts from gridded data in uv domain

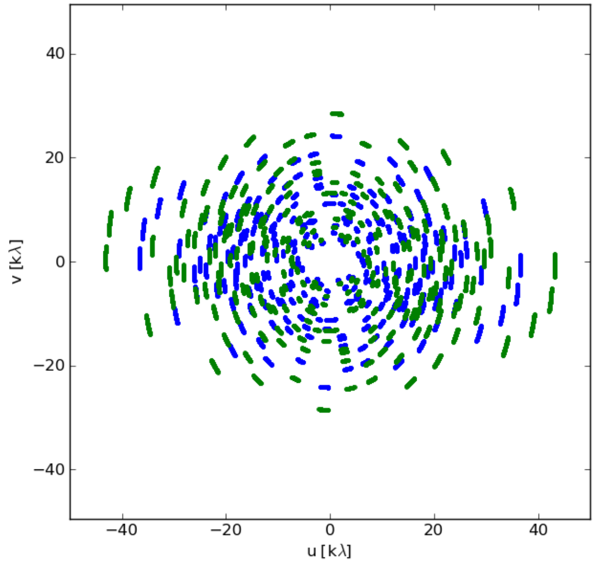
Deconvolution algorithms: Cotton-Schwab



Cotton-Schwab (csclean):

- subtracts truncated PSF in image domain
- major cycle subtracts from full visibilities
- significant I/O per major cycle

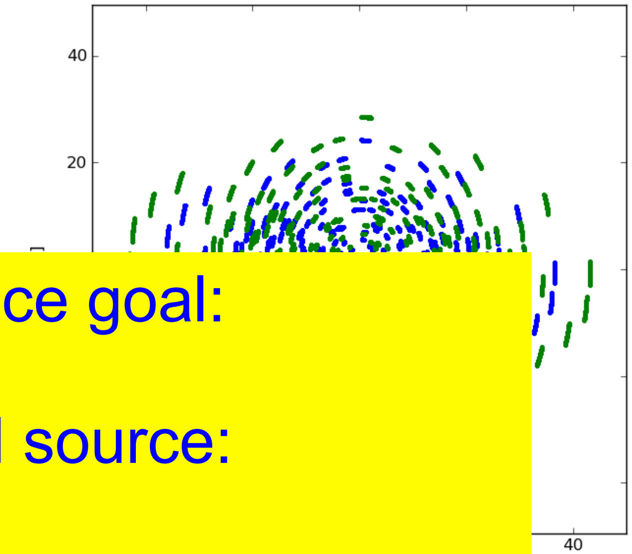
Dirty Beam Shape and Weighting



- Each visibility point is given a weight in the imaging step
- First piece: weight given by Tsys, integration time, etc.
- **Natural**
  - Each sample is given the same weight
  - There are many samples at short baselines, so natural weighting will give the largest beam and the best surface brightness sensitivity (and sometimes pronounced wings in the dirty beam)
- **Uniform**
  - each visibility is given a weight inversely proportional to the sample density
  - Weighs down short baselines, long baselines are more pronounced. Best resolution; poorer noise characteristics
- **Briggs (Robust)**
  - A graduated scheme using the parameter *robust;* compromise of noise and resolution
  - In CASA, set *robust* from -2 (~ uniform) to +2 (~ natural)
  - *robust* = 0.5 often a good choice

Taper: additional weight function to be applied (typically a Gaussian to suppress the weights of the outer visibilities – be careful, however, not to substantially reduce the collecting area)

Dirty Beam Shape and Weighting



- Each visibility point is given a weight in the imaging step
- First piece: weight given by Tsys, integration time, etc.
- **Natural**

Adjust the weighting to match your science goal:

→ Detection experiment/weak extended source: **natural** (maybe even with a taper)

→ Finer detail of strong sources: **robust** or even **uniform**

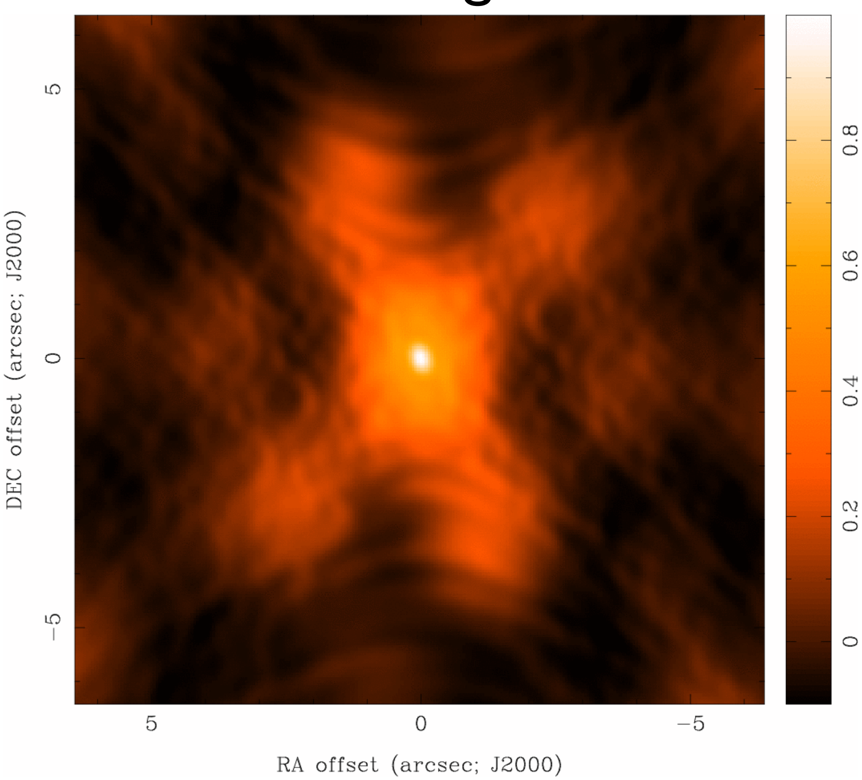
- **Briggs (Robust)**
  - A graduated scheme using the parameter *robust;* compromise of noise and resolution
  - In CASA, set *robust* from -2 (~ uniform) to +2 (~ natural)
  - *robust* = 0 often a good choice

NRAO

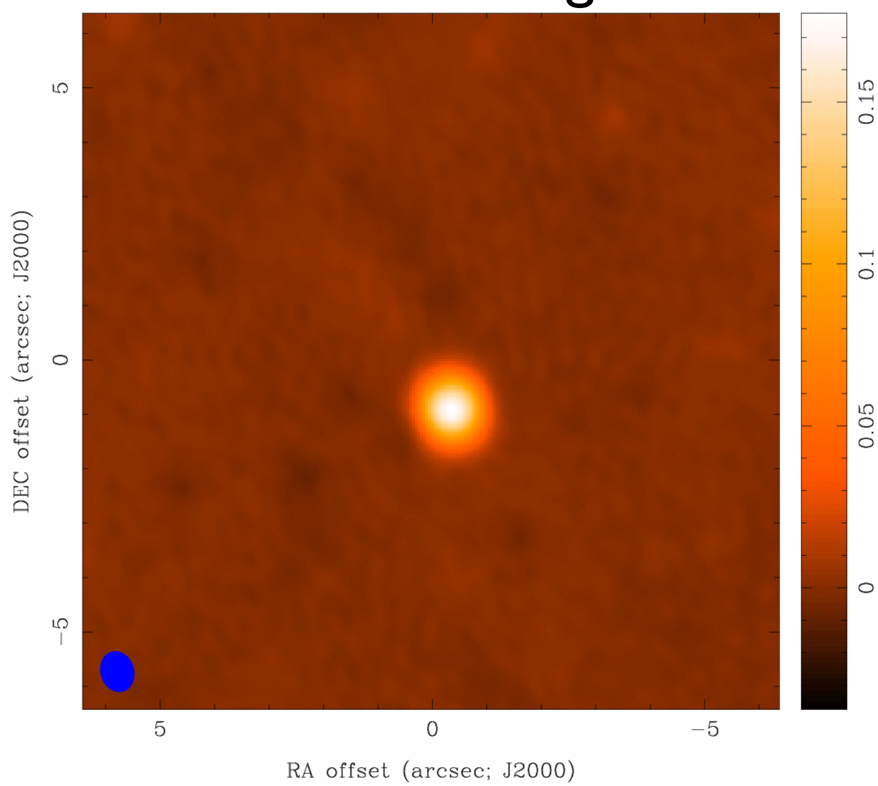
Taper: additional weight function to be applied (typically a Gaussian to suppress the weights of the outer visibilities – be careful, however, not to substantially reduce the collecting area)

Imaging Results

Natural Weight Beam

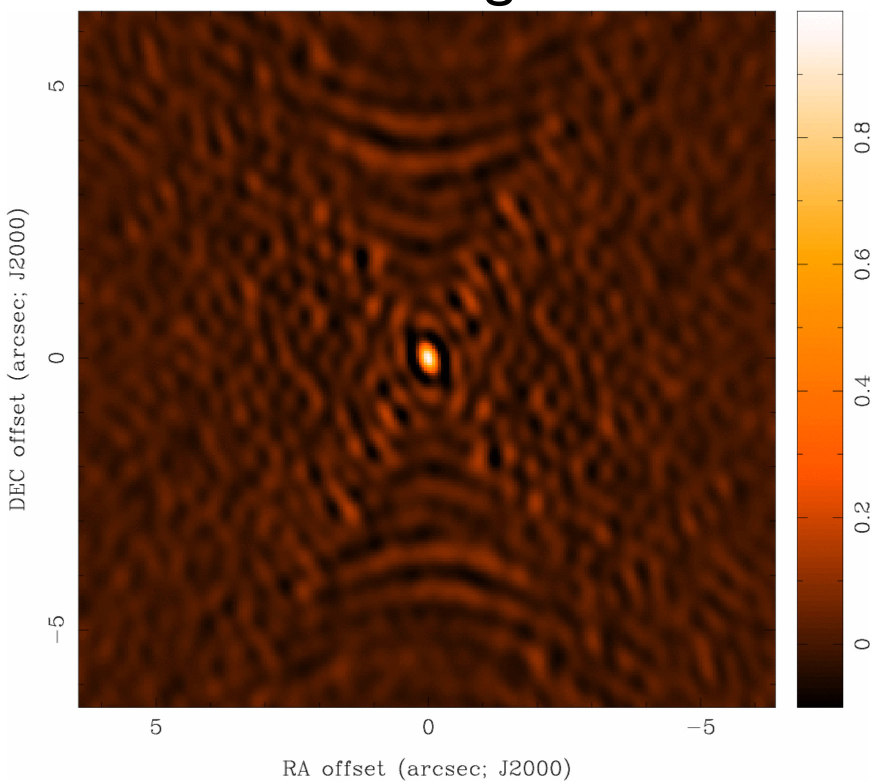


CLEAN image

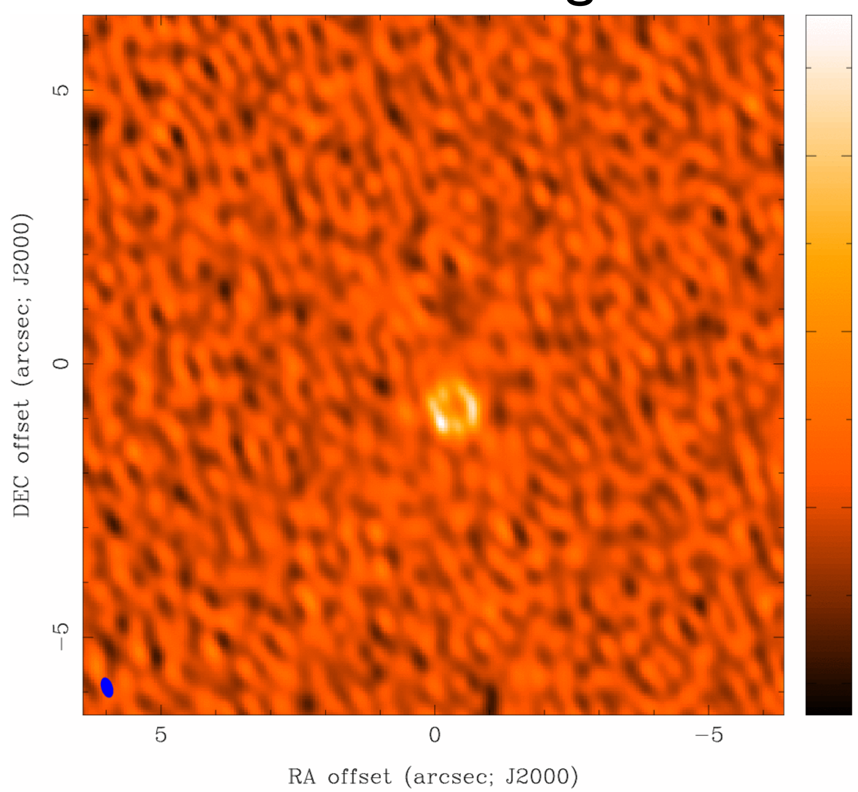


Imaging Results

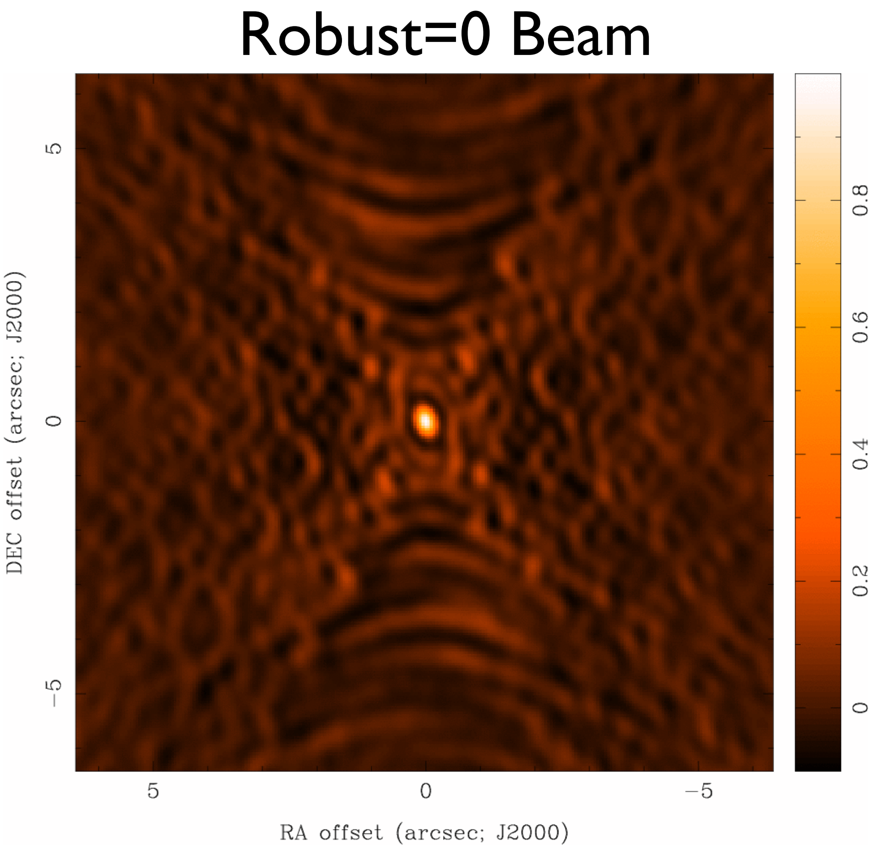
Uniform Weight Beam

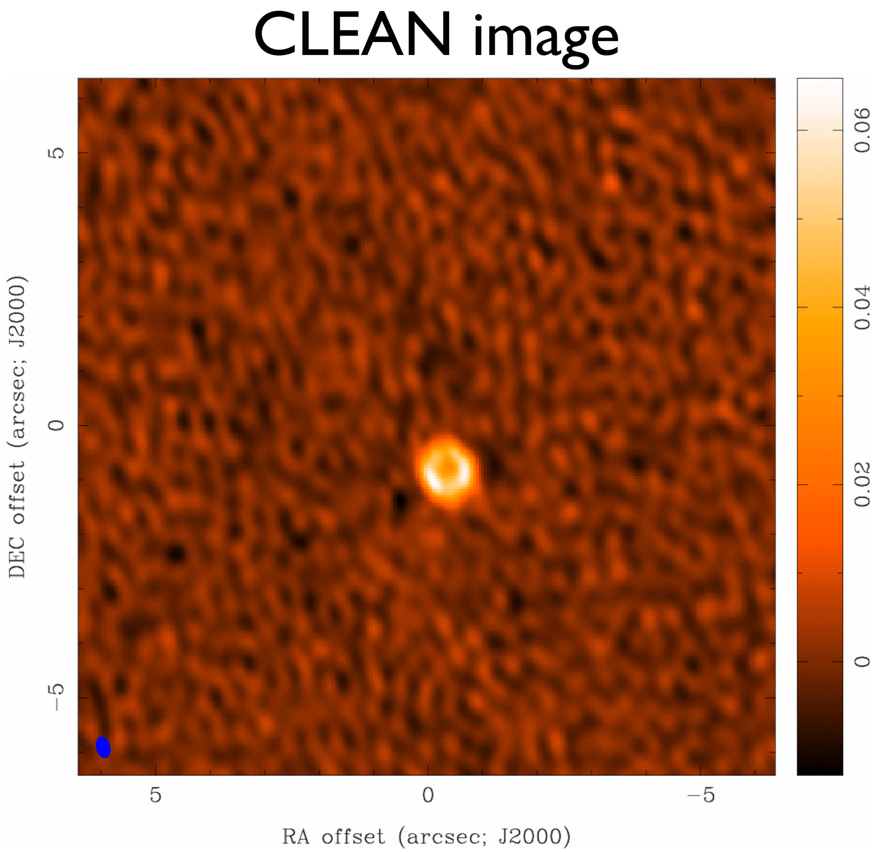


CLEAN image



Imaging Results





CLEAN in CASA

CLEAN is **the** imaging task in CASA. It:

- takes the calibrated visibilities
- grids them on the UV-plane
- performs the FFT to a dirty image
- deconvolves the image
- restores the image from clean table and residual

Modes/Capabilities:

- continuum: incl. multi-frequency synthesis (radial extend of each visibility due to bandwidth), and Taylor term expansion (to derive spectral index and curvature
- spectral line: data cubes (many planes) grids in velocity space, takes account of Doppler shift of line
- mosaicking: combine multiple pointings to single image
- w-projection/faceting for images beyond the half-power point
- outlier fields to deconvolve strong sources in primary beam sidelobes
- multiscale cleaning
- primary beam correction

NRAO

CLEAN in CASA:

```
CASA <3>: inp clean
--------> inp(clean)
#  clean :: Invert and deconvolve images with selected algorithm
vis                 =         ''        #  Name of input visibility file
imagename           =         ''        #  Pre-name of output images
outlierfile         =         ''        #  Text file with image names, sizes, centers for
                                        #   outliers
field               =         ''        #  Field Name or id
spw                 =         ''        #  Spectral windows e.g. '0~3', '' is all
selectdata          =       True        #  Other data selection parameters
     timerange      =         ''        #  Range of time to select from data
     uvrange        =         ''        #  Select data within uvrange
     antenna        =         ''        #  Select data based on antenna/baseline
     scan           =         ''        #  Scan number range
     observation    =         ''        #  Observation ID range
     intent         =         ''        #  Scan Intent(s)

mode                =      'mfs'        #  Spectral gridding type (mfs, channel, velocity,
                                        #   frequency)
gridmode            =         ''        #  Gridding kernel for FFT-based transforms,
                                        #   default='' None
niter               =        500        #  Maximum number of iterations
gain                =        0.1        #  Loop gain for cleaning
threshold           =   '0.0mJy'        #  Flux level to stop cleaning, must include
                                        #   units: '1.0mJy'
psfmode             =    'clark'        #  Method of PSF calculation to use during minor
                                        #   cycles
imagermode          =  'csclean'        #  Options: 'csclean' or 'mosaic', '', uses
                                        #   psfmode
     cyclefactor    =        1.5        #  Controls how often major cycles are done. (e.g.
                                        #   5 for frequently)
     cyclespeedup   =         -1        #  Cycle threshold doubles in this number of
                                        #   iterations

multiscale          =         []        #  Deconvolution scales (pixels); [] = standard
                                        #   clean
interactive         =      False        #  Use interactive clean (with GUI viewer)
mask                =         []        #  Cleanbox(es), mask image(s), region(s), or a
                                        #   level
imsize              = [256, 256]        #  x and y image size in pixels.  Single value:
                                        #   same for both
cell                = ['1.0arcsec']     #  x and y cell size(s). Default unit arcsec.
phasecenter         =         ''        #  Image center: direction or field index
restfreq            =         ''        #  Rest frequency to assign to image (see help)
stokes              =        'I'        #  Stokes params to image (eg I,IV,IQ,IQUV)
weighting           =  'natural'        #  Weighting of uv (natural, uniform, briggs, ...)
uvtaper             =      False        #  Apply additional uv tapering of visibilities
modelimage          =         ''        #  Name of model image(s) to initialize cleaning
restoringbeam       =       ['']        #  Output Gaussian restoring beam for CLEAN image
pbcor               =      False        #  Output primary beam-corrected image
minpb               =        0.2        #  Minimum PB level to use
usescratch          =      False        #  True if to save model visibilities in
                                        #   MODEL_DATA column
allowchunk          =      False        #  Divide large image cubes into channel chunks
                                        #   for deconvolution
```

Basic Image Parameters: Pixel Size and Image Size

- **Pixel size**
  - should satisfy $\Delta x < 1/2\, u_{max}$, $\Delta y < 1/2\, v_{max}$ (Nyquist)
  - in practice, ~5 pixels across the main lobe of the beam works better
- **Image size**
  - Consider FWHM of primary beam (e.g. ~ 20" at Band 7)
  - Be aware that sensitivity is not uniform across the primary beam (may need primary beam correction)
  - Use mosaicking to image larger targets
  - Not restricted to powers of 2; CASA performs best at given image sizes, rule of thumb: $2^n * 10$
  - If there are bright sources in the sidelobes, they will throw sidelobes onto the image, so image large to be able to clean them out, or use outlierfile to specify the positions of outlier fields ("peeling")

Output of CLEAN

Minimally:

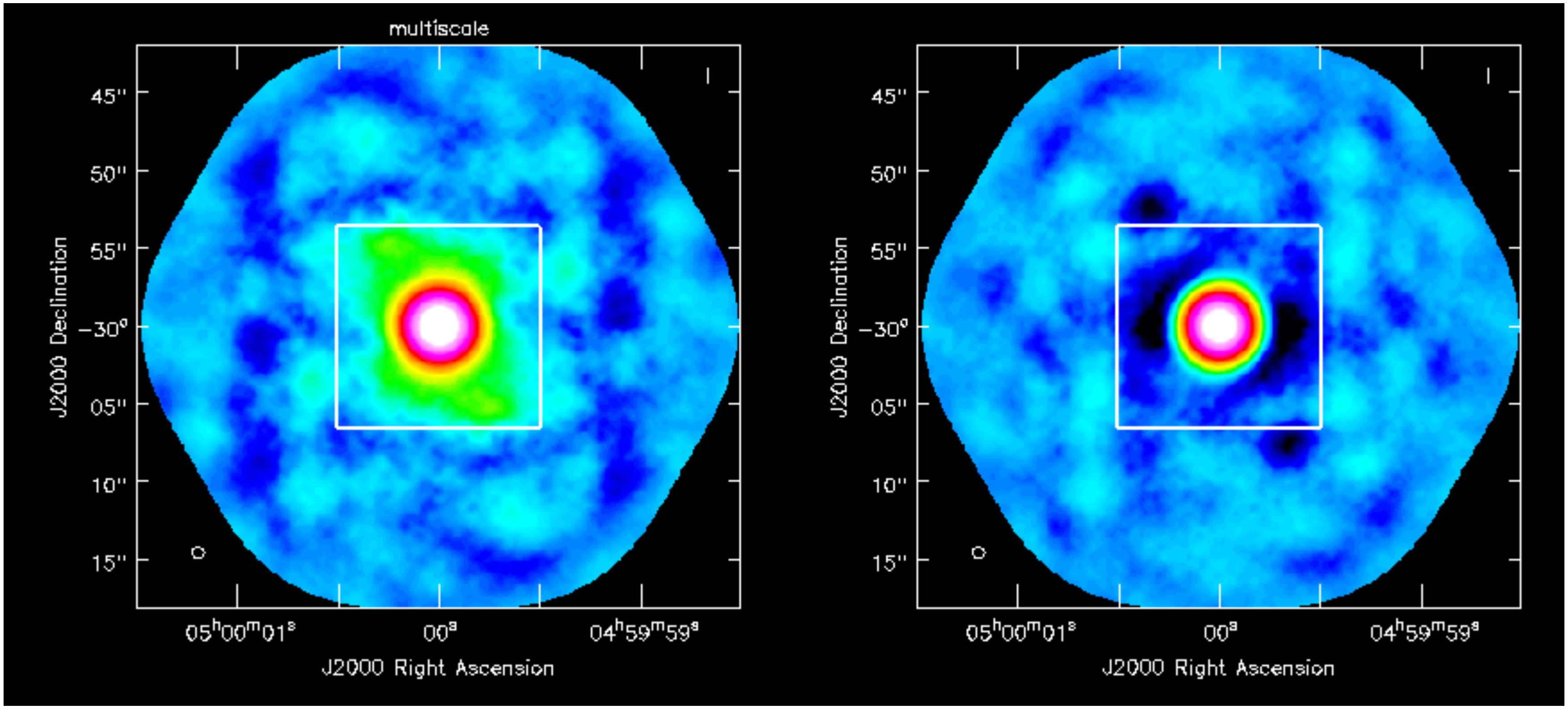
- my_image.flux — Relative sky sensitivity
- my_image.image — Cleaned and restored image (Jy/clean beam)
- my_image.mask — Clean “boxes”
- my_image.model — Clean components (Jy/pixel)
- my_image.psf — Dirty beam
- my_image.residual — Residual (Jy/dirty beam)

If CLEAN is started again with same image name, it will try to continue deconvolution from where it left off. Make sure this is what you want. If not, give a new name or remove existing files with rmtables(‘my_image.*’)

Do NOT do CTRL+C as it could corrupt your MS when it touches the visibilities in a major cycle. (Always good practice to make a backup copy of your MS first.)

Multi-scale CLEAN

multi-scale "classic" scale

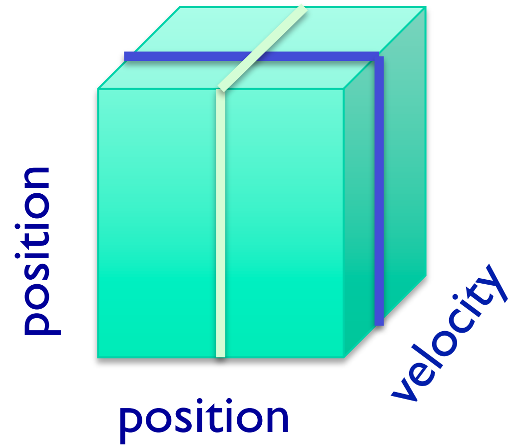



```
multiscale          = [0, 5, 12, 24, 50] #  Deconvolution scales (pixels); [] =
                                         #  standard clean
```

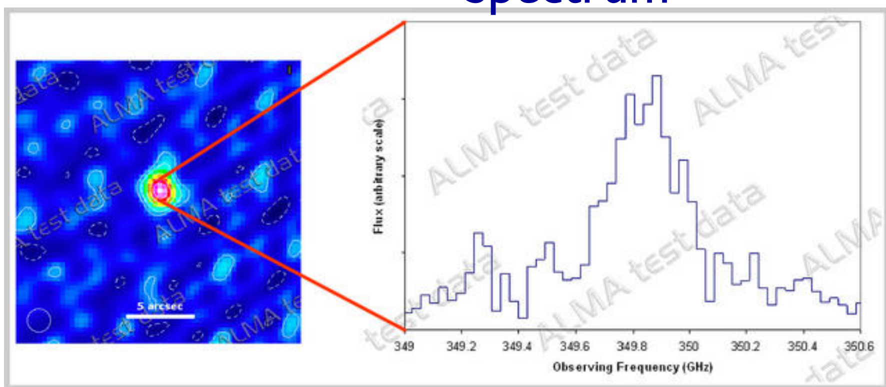
Instead of delta functions, one can use extended clean components to better match emission scales (multiscales, typically paraboloids)

Pick delta function, half the largest emission and a few in between

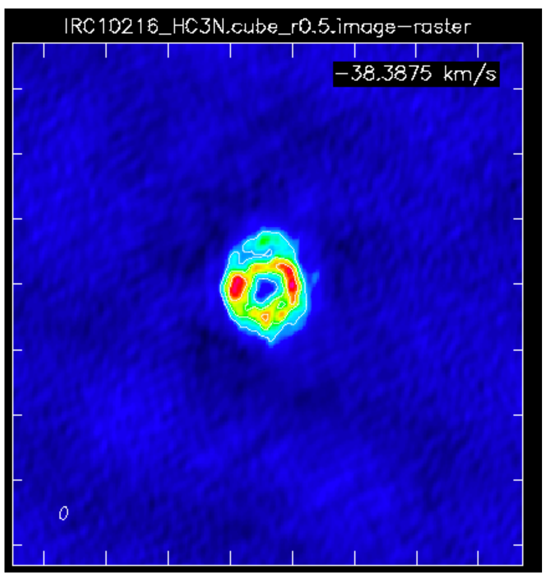
Imaging spectral lines



- Spectrum

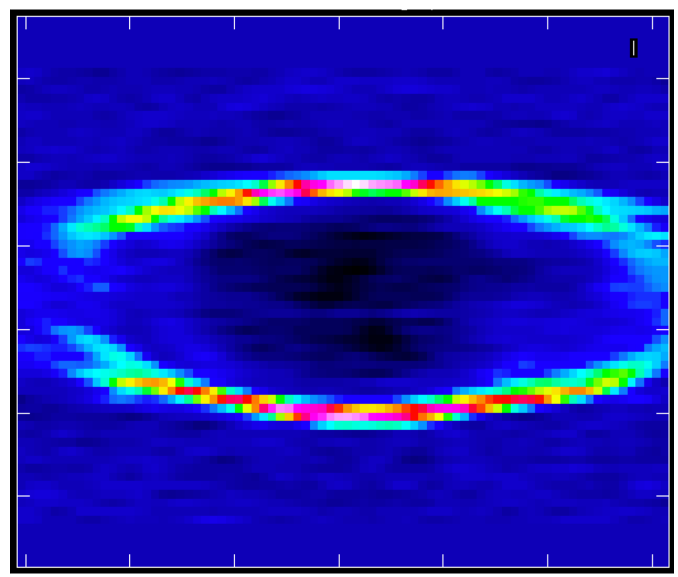


- Channel map



Fixed velocity, polarization, etc.

- Position-velocity map



One fixed position, polarization, etc.

Imaging spectral lines

```
mode            =  'velocity'       #  Spectral gridding type (mfs, channel,
                                    #   velocity, frequency)
nchan           =        100        #  Number of channels (planes) in output
                                    #   image; -1 = all
start           =  '300km/s'        #  Velocity of first channel: e.g
                                    #   '0.0km/s'(''=first channel in first
                                    #   SpW of MS)
width           =   '10km/s'        #  Channel width e.g '-1.0km/s'
                                    #   (''=width of first channel in first
                                    #   SpW of MS)
interpolation   =   'linear'        #  Spectral interpolation (nearest,
                                    #   linear, cubic).
resmooth        =      False        #  Re-restore the cube image to a common
                                    #   beam when True
chaniter        =      False        #  Clean each channel to completion
                                    #   (True), or all channels each cycle
                                    #   (False)
outframe        =     'LSRK'        #  Velocity reference frame of output
                                    #   image; '' =input
veltype         =    'radio'        #  Velocity definition of output image
restfreq              = '115.271201800GHz' #  Rest frequency to assign to image (see help)
```

mode="velocity"

→ Set the dimensions of the cube

→ Set Rest frequency

→ Set Velocity Frame (LSRK, BARY, …)

→ Set Doppler definition (optical/radio) (if in doubt use radio as channels are constant frequency width)

Clean will calculate the Doppler corrections for you! No need to realign beforehand. (but **cvel** will do it for you if needed, e.g. when self-calibrating)

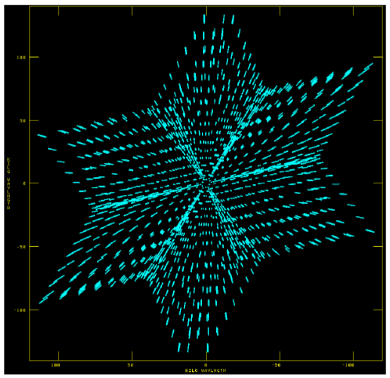
Imaging spectral lines: continuum subtraction

- Generally would like to subtract continuum emission (we will see how to identify line-free channels in hands-on session)
- Use `uvcontsub` to do the subtraction in uv plane.

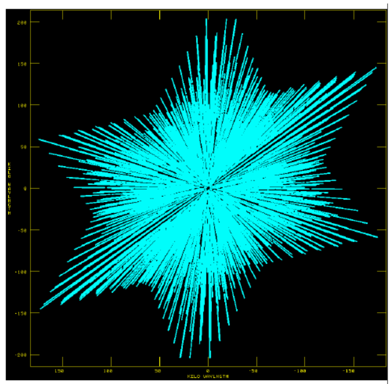
```
CASA <11>: inp
---------> inp()
#  uvcontsub :: Continuum fitting and subtraction in the uv plane
vis                 = 'ngc3256_co.ms'   #  Name of input MS.  Output goes to vis + ".contsub"
field               =        ''         #  Select field(s) using id(s) or name(s)
fitspw              = '0:20~53;71~120'  #  Spectral window:channel selection for fitting the continuum
combine             =        ''         #  Data axes to combine for the continuum estimation (none, or spw and/or scan)
solint              =     'int'         #  Continuum fit timescale (int recommended!)
fitorder            =         0         #  Polynomial order for the fits
spw                 =        ''         #  Spectral window selection for output
want_cont           =     False         #  Create vis + ".cont" to hold the continuum estimate.
async               =     False         #  If true the taskname must be started using uvcontsub(...)
```

Continuum Imaging

- Multi-scale Multi-Frequency Taylor Term expansion

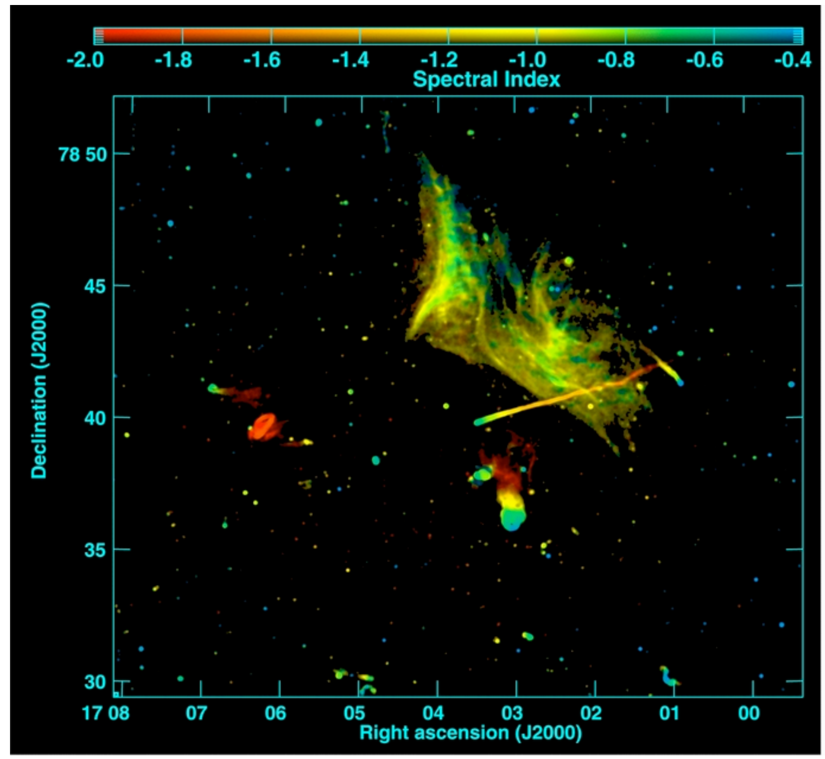


Narrow BW



wide BW
(better uv-coverage)

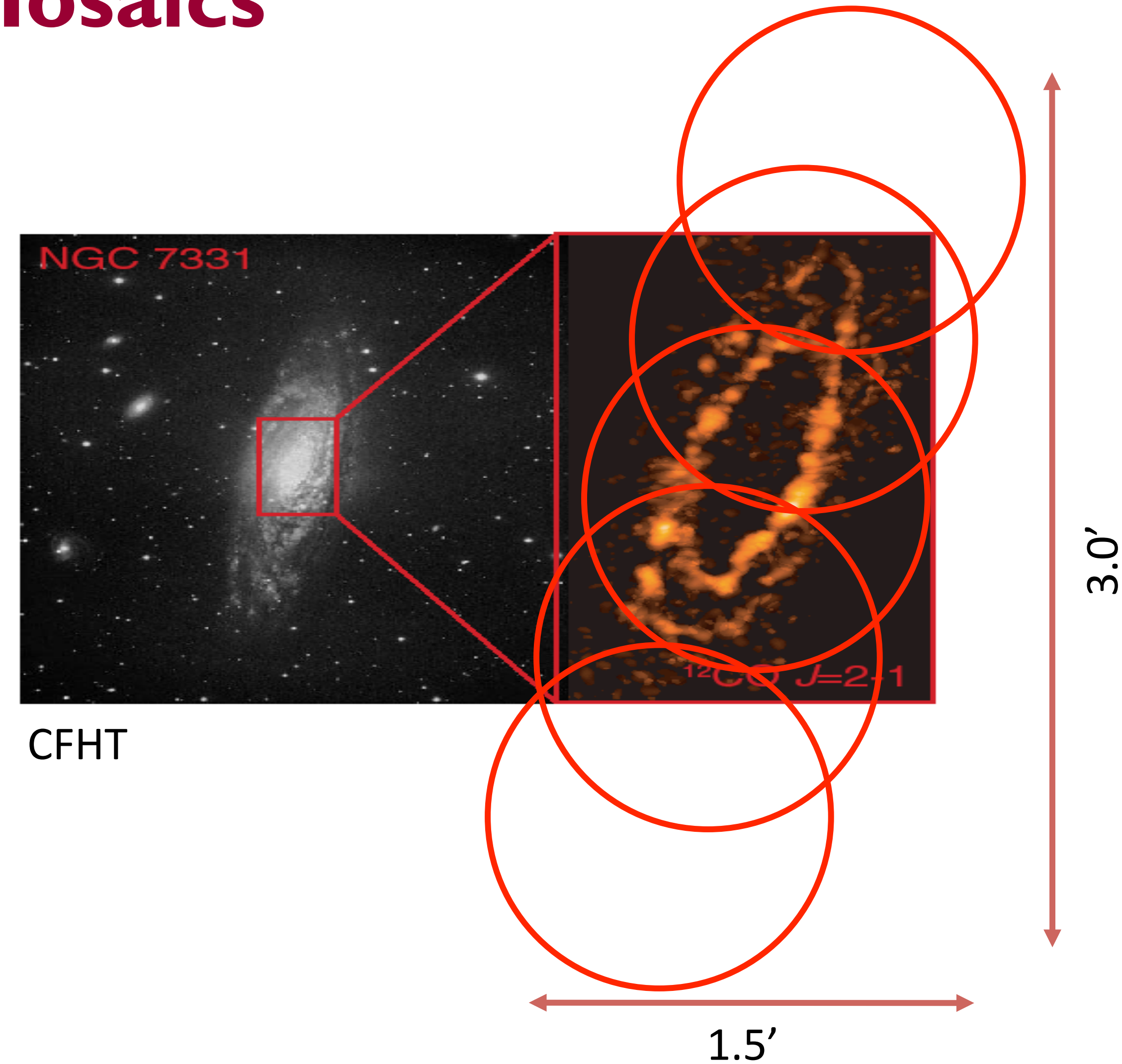
- Plus spectral index:



Abell 2256; Owen et al. (2014)

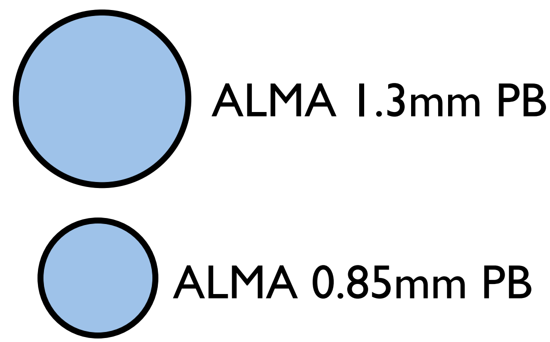
- MFS (mode mfs)
  - nterm=2 compute spectral index, 3 for curvature etc.
  - needed for bandwidths ~5% or more (S/N dependent) (most VLA continuum)
  - tt0 average intensity, tt1 alpha*tt0, alpha images output
  - takes at least nterms longer (image size dependent)

Mosaics



Example: SMA 1.3 mm observations: 5 pointings

- Primary beam ~1'
- Resolution ~3"



Petitpas et al.

Imaging mosaics

```
imagermode      =   'mosaic'   #  Options: 'csclean' or 'mosaic', '', uses psfmode
     mosweight  =      False   #  Individually weight the fields of the mosaic
     ftmachine  =       'ft'   #  Gridding method for the image
     scaletype  =    'SAULT'   #  Controls scaling of pixels in the image plane. default='SAULT'; example:
                               #   scaletype='PBCOR' Options: 'PBCOR','SAULT'
     cyclefactor  =      1.5   #  Controls how often major cycles are done. (e.g. 5 for frequently)
     cyclespeedup =       -1   #  Cycle threshold doubles in this number of iterations
     flatnoise    =     True   #  Controls whether searching for clean components is done in a constant noise
                               #   residual image (True) or in an optimal signal-to-noise residual image
                               #   (False)
```

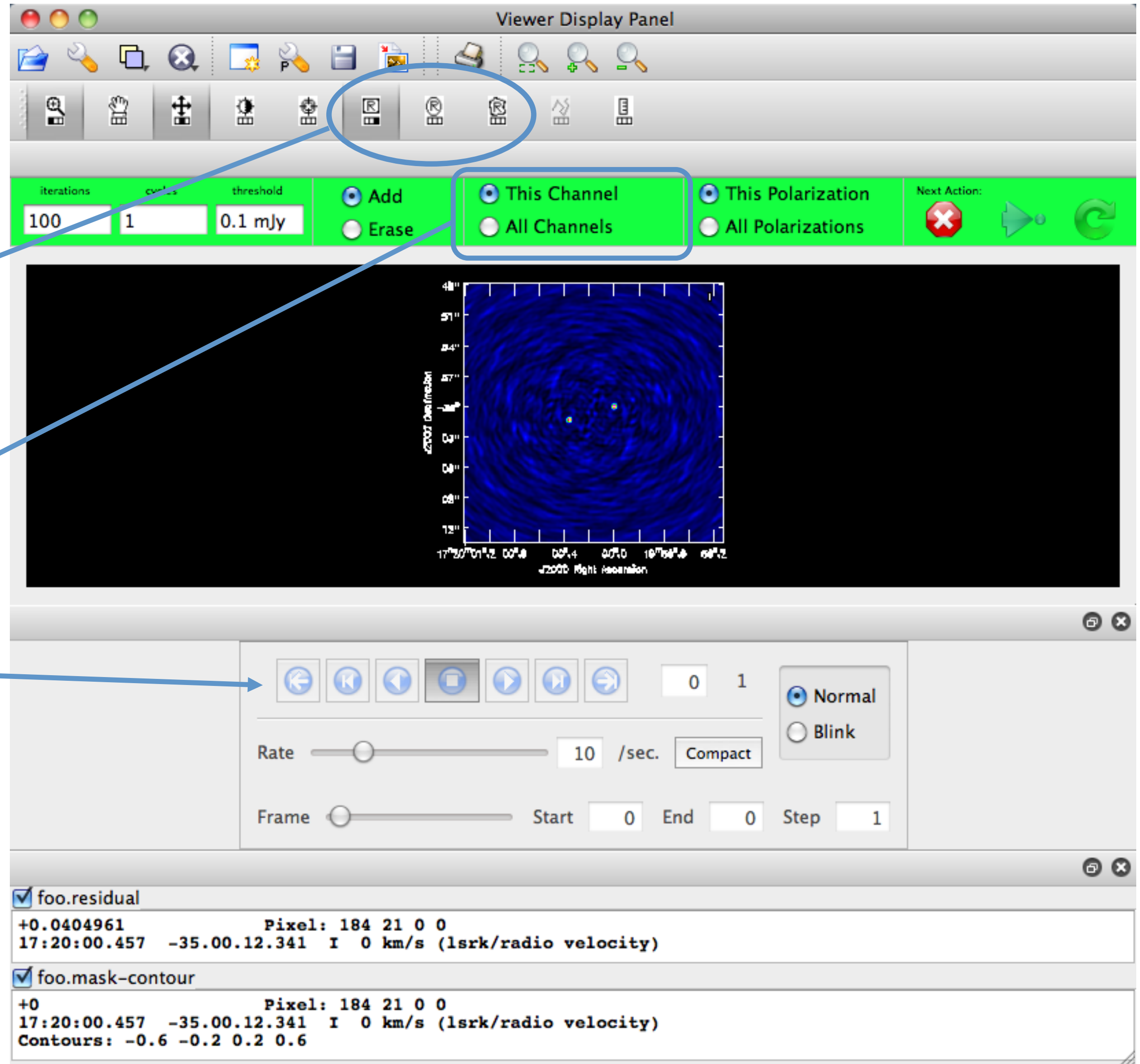
ftmachine = “mosaic” : add in uv plane and invert together, Use *csclean* for deconvolution.

ftmachine = “ft” : shift and add in image plane

There’s a tool (“ia.linearmosaic”) to linear mosaic after cleaning each pointing and to stitch all pointings together entirely in the image domain

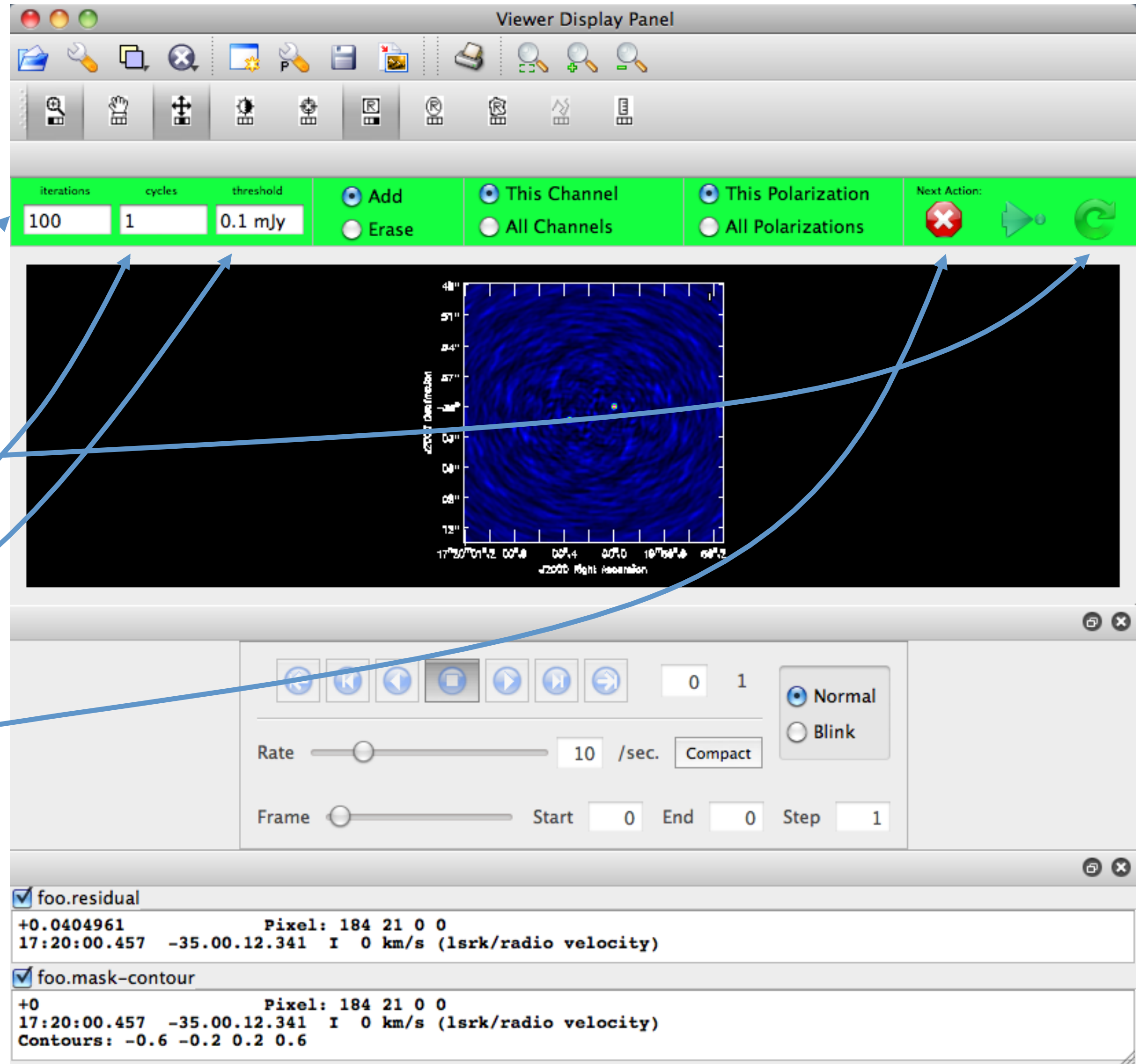
Interactive CLEAN

- residual image in viewer
- define a mask with defining a mouse button on shape type
- define the same mask for all channels
- or iterate through the channels with the tape deck and define separate masks



Interactive CLEAN

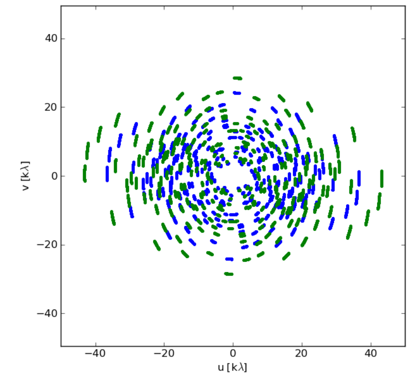
- perform N iterations
- and return – every time the residual is displayed is a major cycle
- continue until #cycles or threshold reached, or user stop

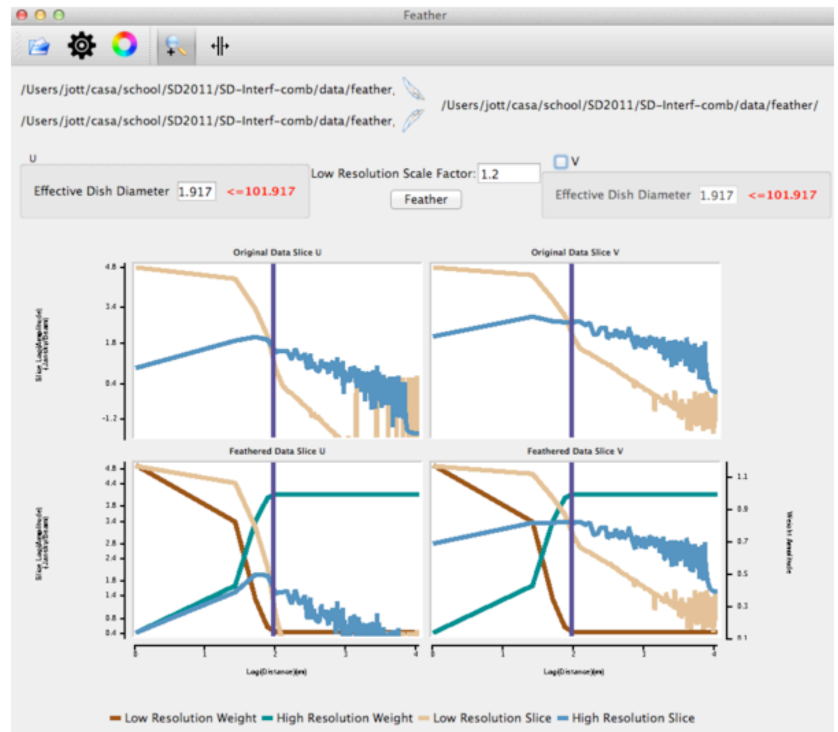


Combining with single-dish or other interferometric maps

- If you have only images:
  - feather (or “casafeather”)
- If you have an image and an MS:
  - use CLEAN with the image as “modelimage”
  - and/or feather
- If you have multiple MS plus an image:
  - Same as above, input to clean will be all the MS.

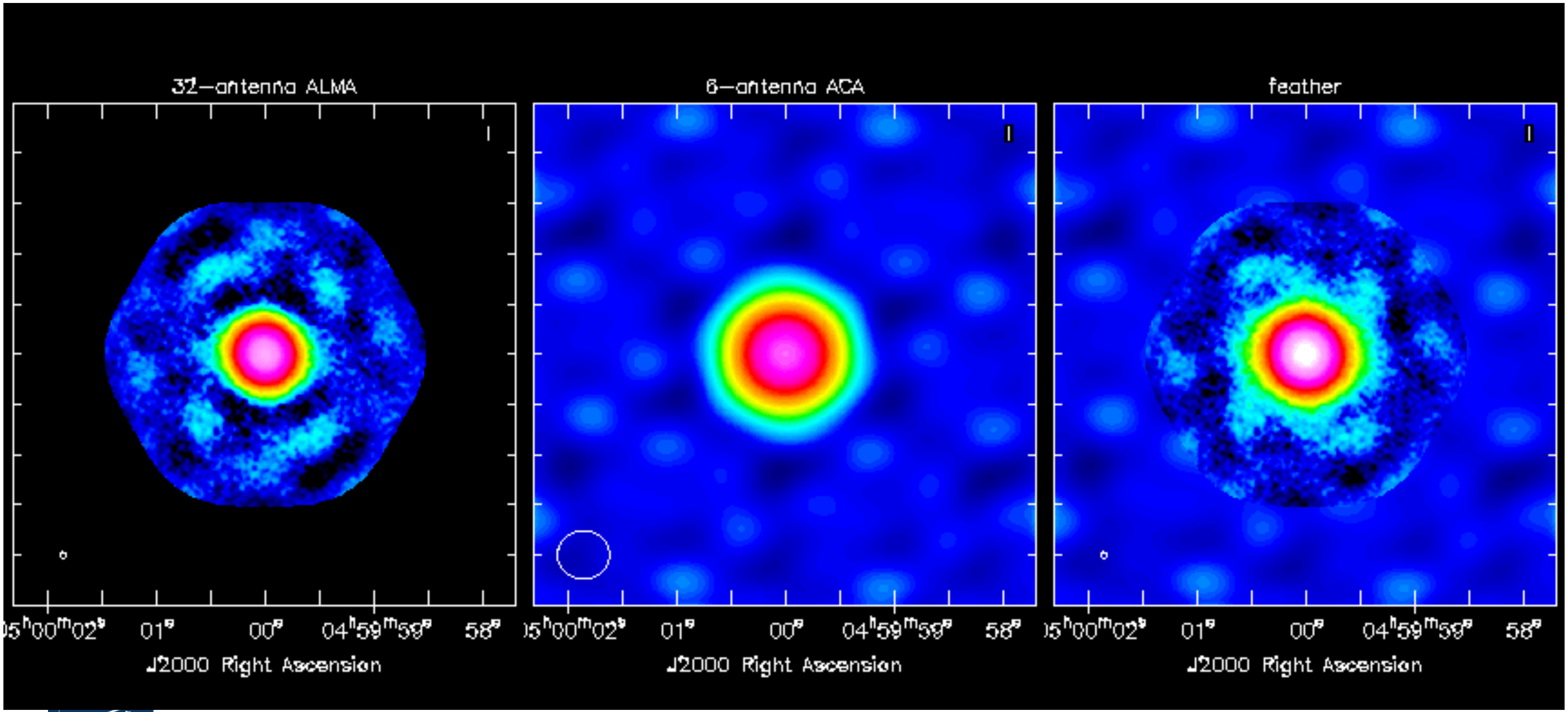
See the M100 CASAGuide for more details: https://casaguides.nrao.edu/index.php?title=M100_Band3





Combining with other data: feather

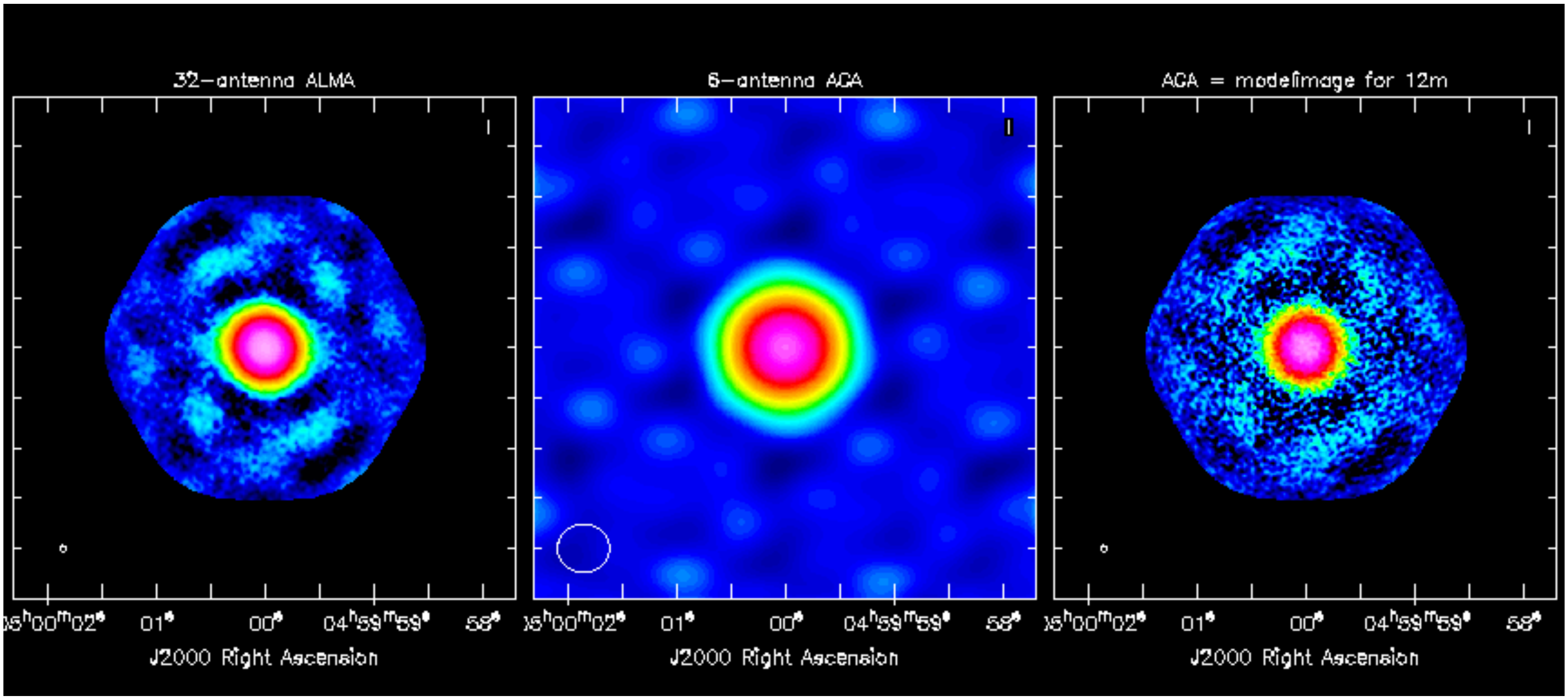
```
#  feather :: Combine two images using their Fourier transforms
imagename          =        ''         #  Name of output feathered image
highres            =        ''         #  Name of high resolution (interferometer) image
lowres             =        ''         #  Name of low resolution (single dish) image
async              =     False         #  If true the taskname must be started using feather(...)
```



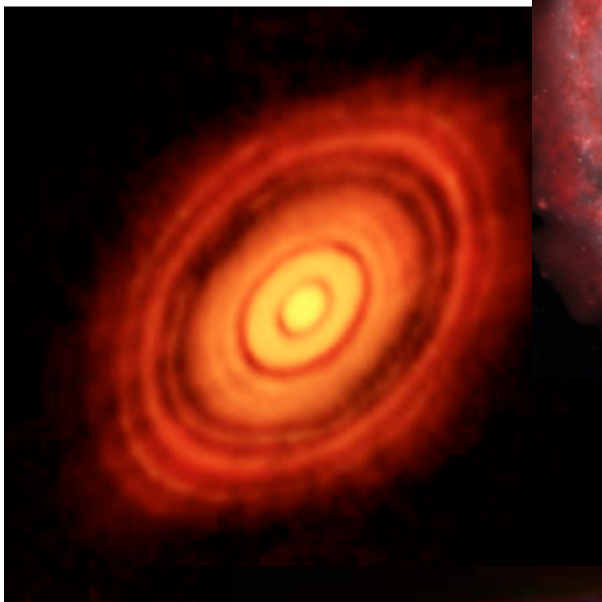
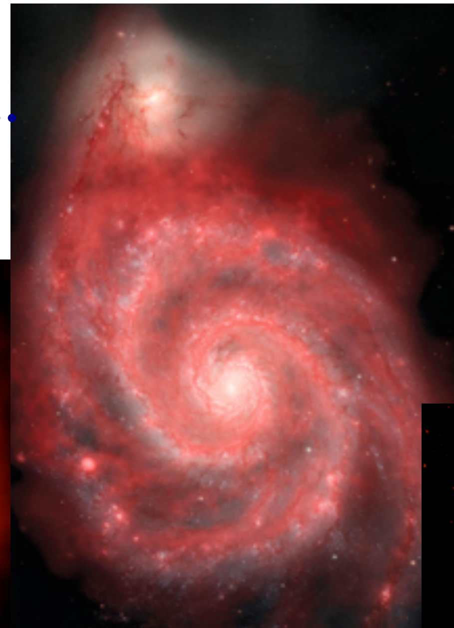
We also have a graphical tool: CASAfeather

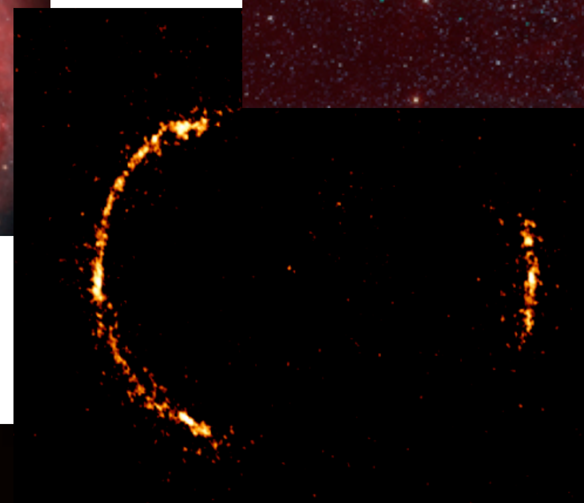
Combining with other data: modelimage

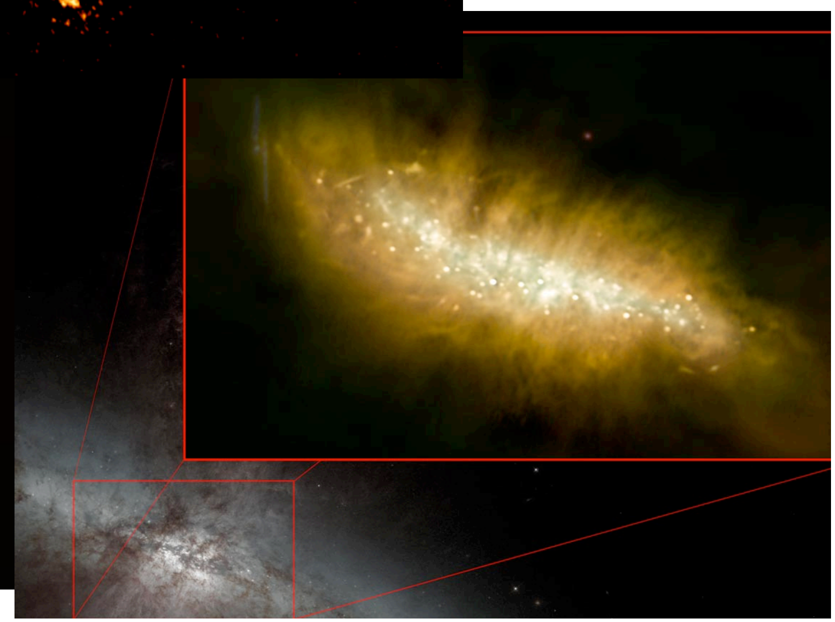
```
--------> inp(clean)
#  clean :: Invert and deconvolve images with selected algorithm
modelimage          =         ''        #  Name of model image(s) to initialize cleaning
```



… some CASA images…





Further reading …

The CASAGuides wiki:

https://casaguides.nrao.edu/index.php/Main_Page

Has more detailed information on calibration and imaging with CASA.