Introduction to Imaging in CASA

Jim Braatz

Based on material from David Wilner, Scott Schnee, Remy Indebetouw

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

NRAO

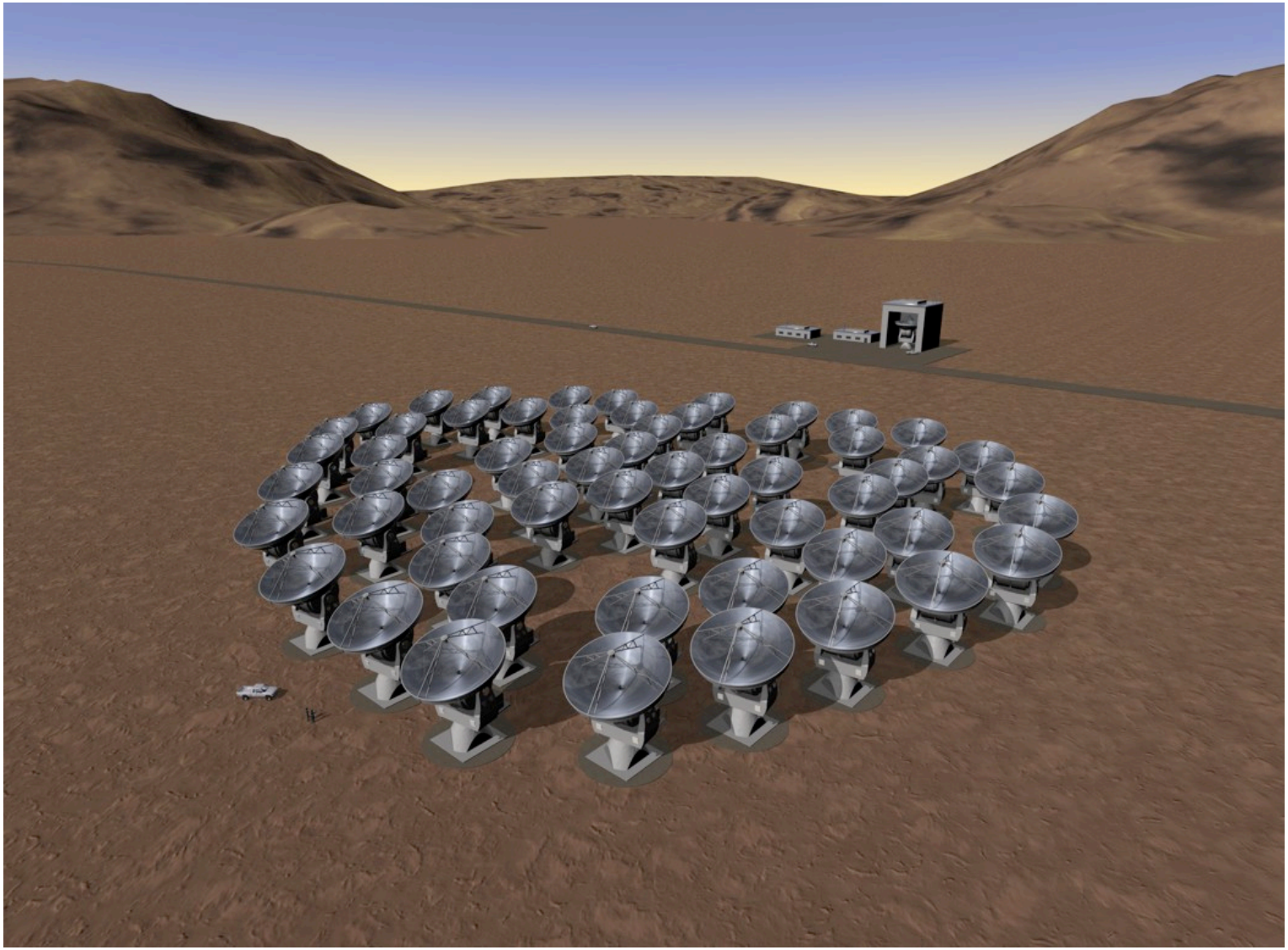
Overview

- 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

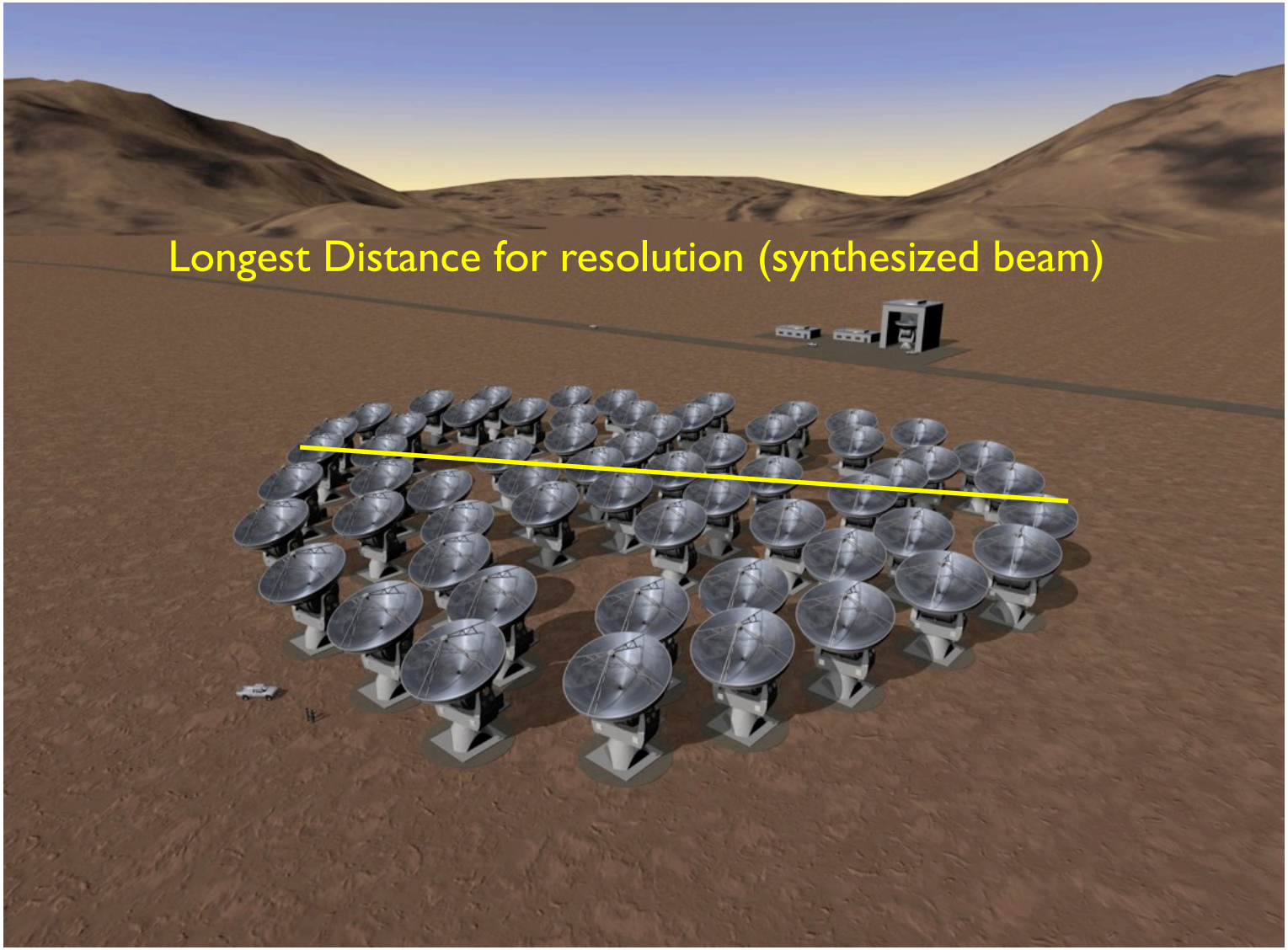
Single dish: diameter is responsible for sensitivity, field of view, resolution

Interferometer: takes this apart



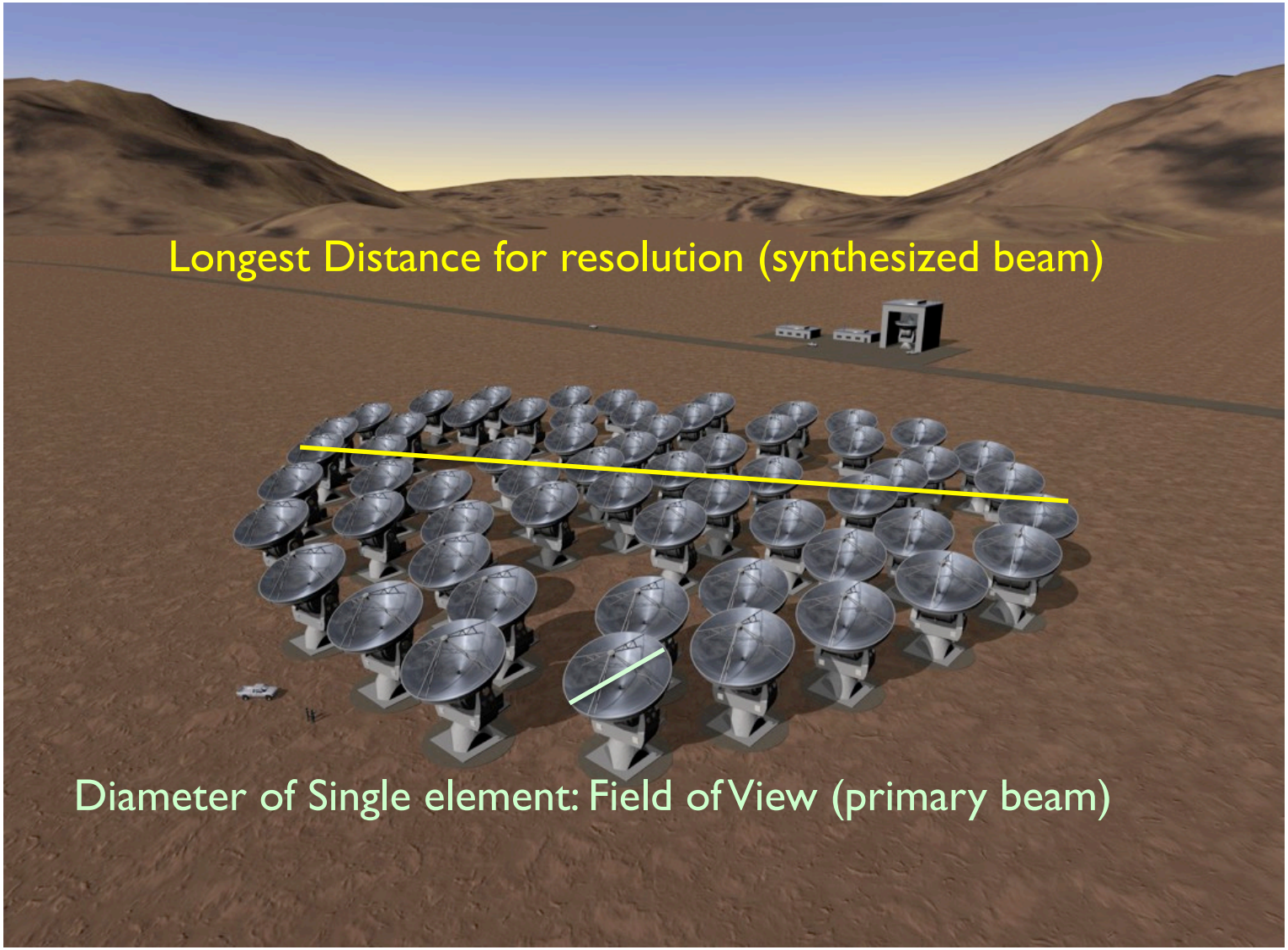
Single dish: diameter is responsible for sensitivity, field of view, resolution

Interferometer: takes this apart



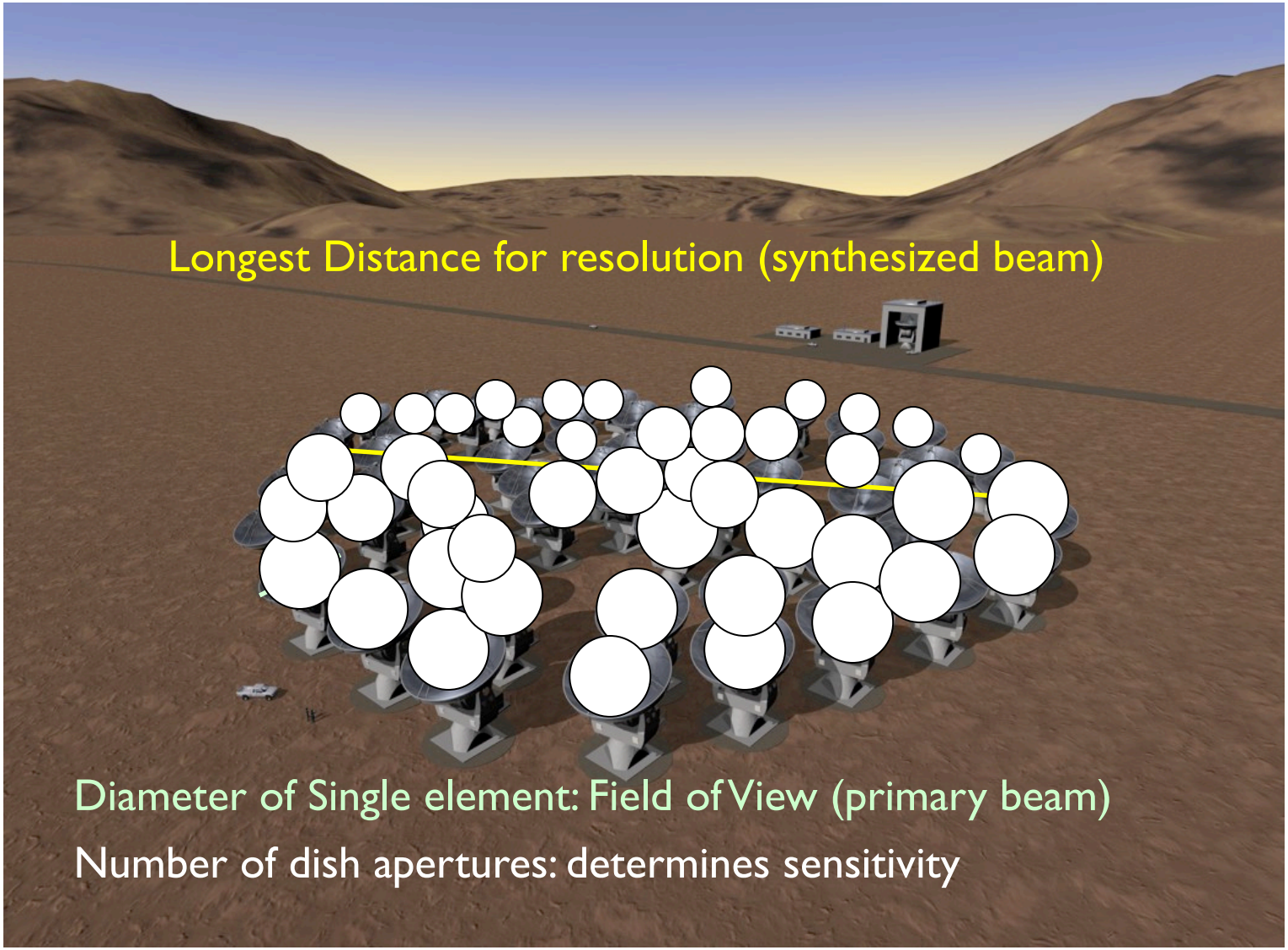
Single dish: diameter is responsible for sensitivity, field of view, resolution

Interferometer: takes this apart



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

Interferometer: takes this apart



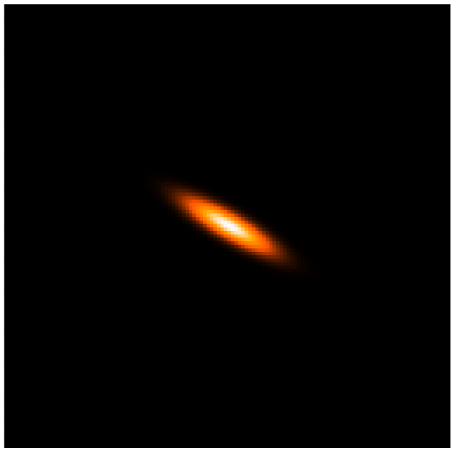
Let's start with a quick review of UV-plane analysis.

Recall from first talk: 2D Fourier Transform Pairs

T(x,y)

Amp{V(u,v)}

elliptical Gaussian

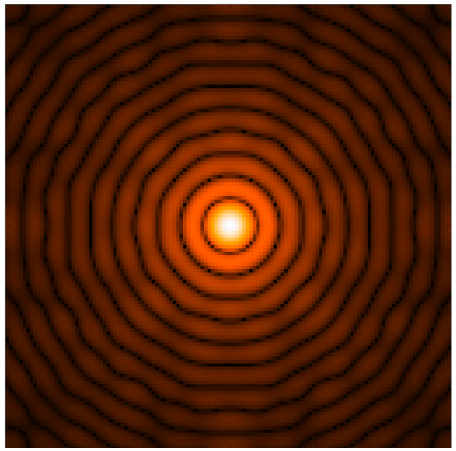


⇌

elliptical Gaussian

Disk

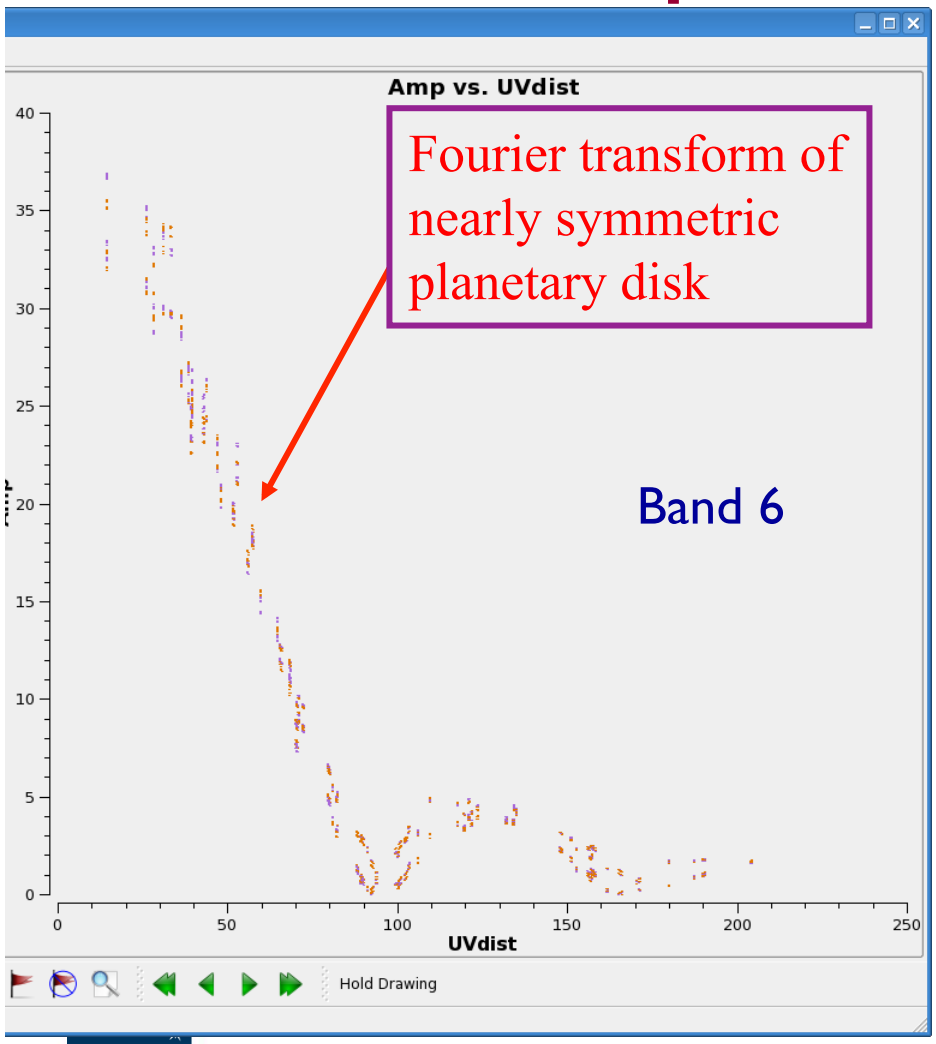
⇌

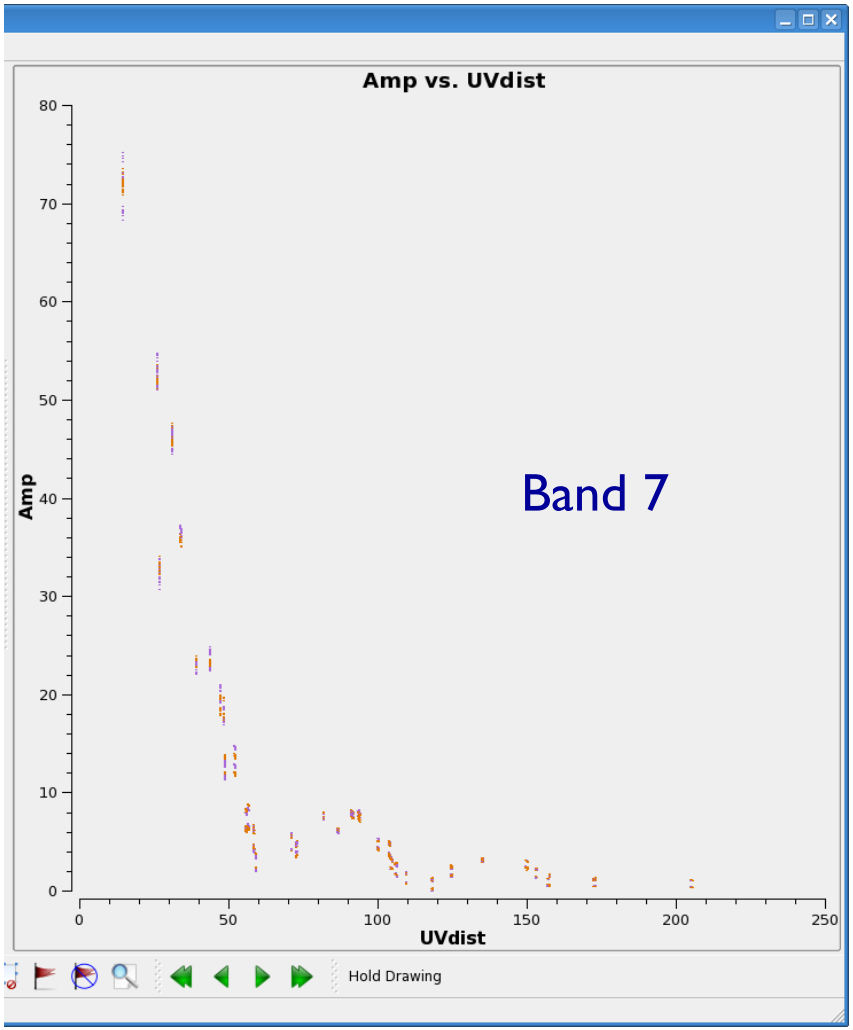


Bessel

sharp edges result in many high spatial frequencies

UV-plane analysis: ALMA observes planetary disk

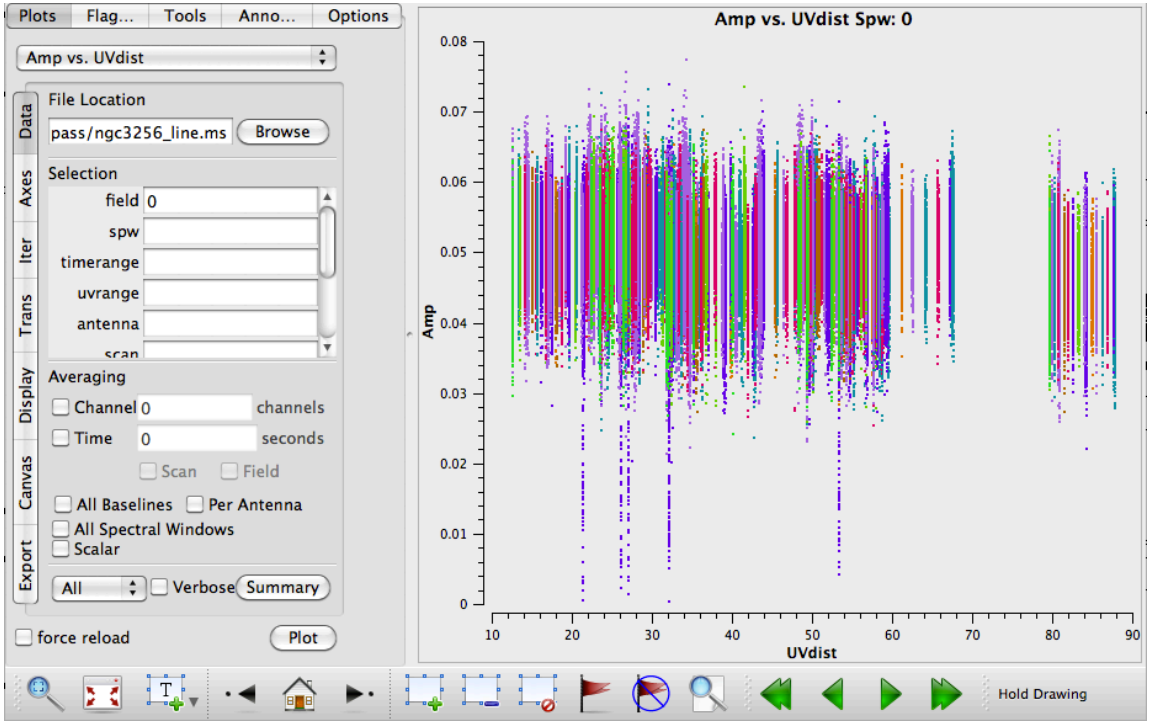




More UV-plane analysis

- A point source (with some bad data)

- By the way, note that you often need to average in time and frequency to "see" your source in the uv-plane

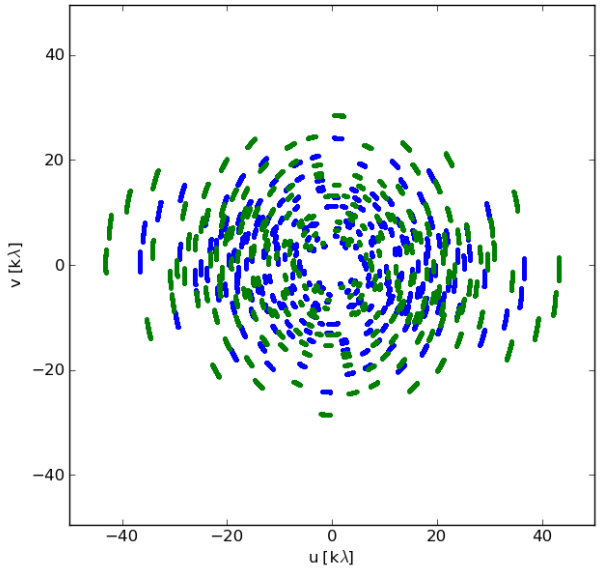


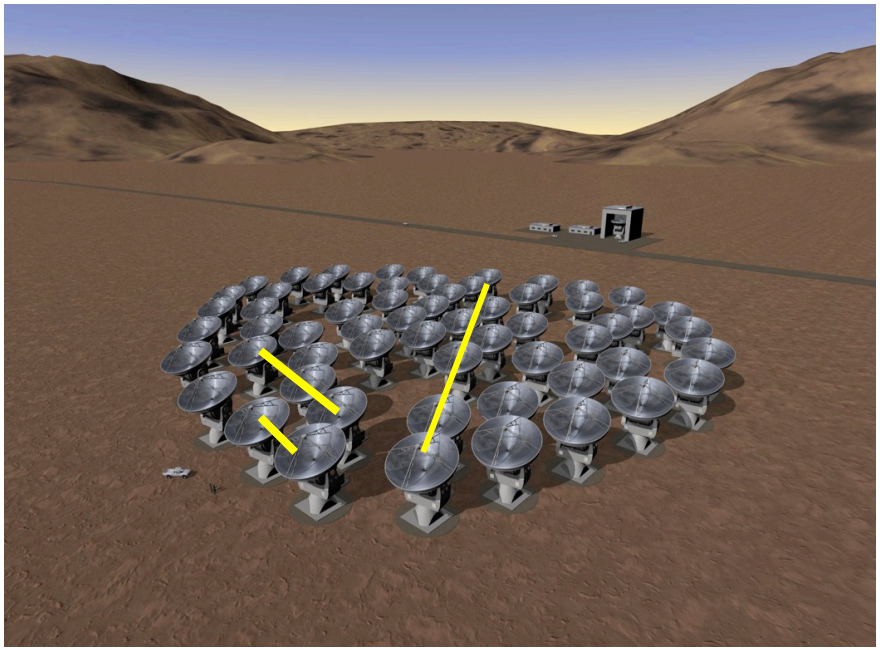
On to Image plane analysis

- UV-plane analysis OK for "simple" sources like point sources and disks
- UV-plane analysis can be very powerful for measuring model parameters in sources with known (usually simple) structure, e.g. measure separation of a pair of point sources.
- But in general, we want to analyze our data in the image plane, being mindful that there are some limitations and caveats from converting to the image plane.

Sampling Function

Interferometers do not measure the entire Fourier/uv domain.
But antenna pairs sample distinct spots: ➔ **imperfect image**





Small uv-distance: short baselines (measure extended emission)
Long uv-distance: long baselines (measure small scale emission)
Orientation of baselines also determine orientation of points in the uv-plane

NRAO

Dirty Images from a Dirty Beam

- We sample Fourier domain at discrete points

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

- the inverse Fourier transform is

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

- the convolution theorem tells us

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

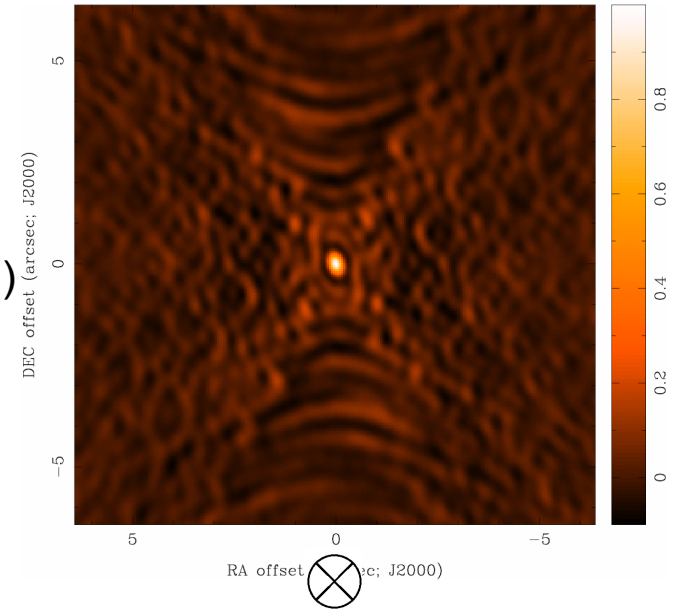
  where $b(x,y) = FT^{-1}\{B(u,v)\}$ (the point spread function)

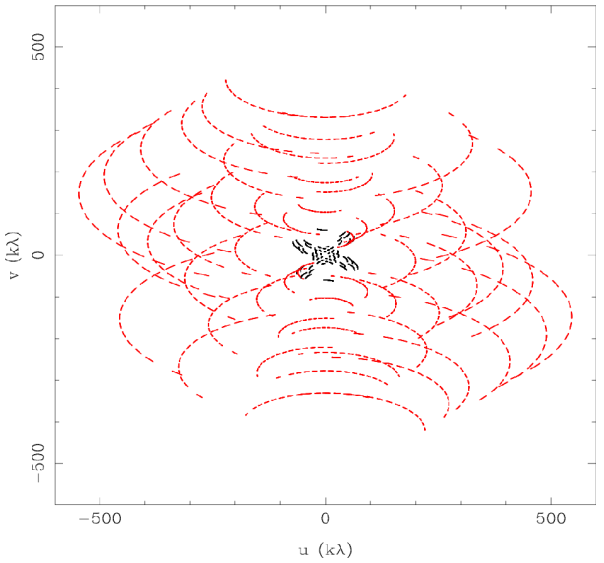
Fourier transform of sampled visibilities yields the true sky brightness convolved with the point spread function

(the "dirty image" is the true image convolved with the "dirty beam")

Dirty Beam and Dirty Image

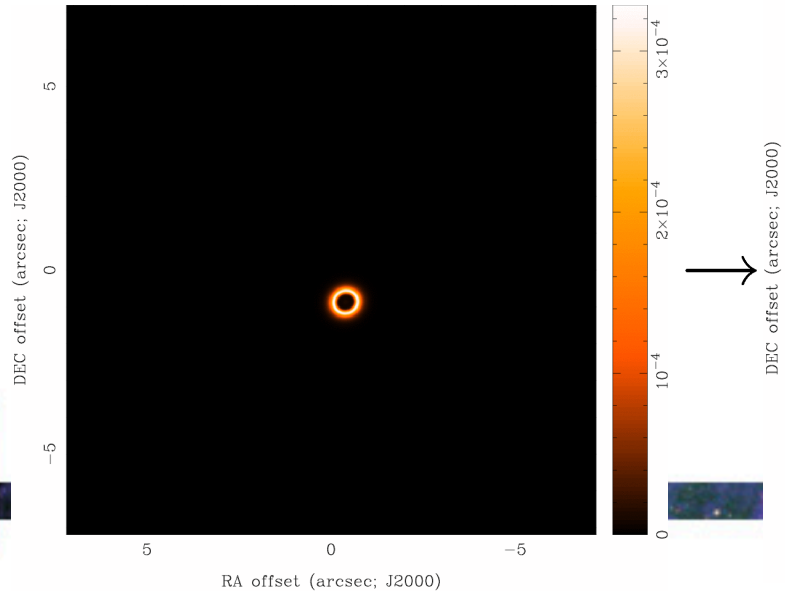
b(x,y)
(dirty beam)

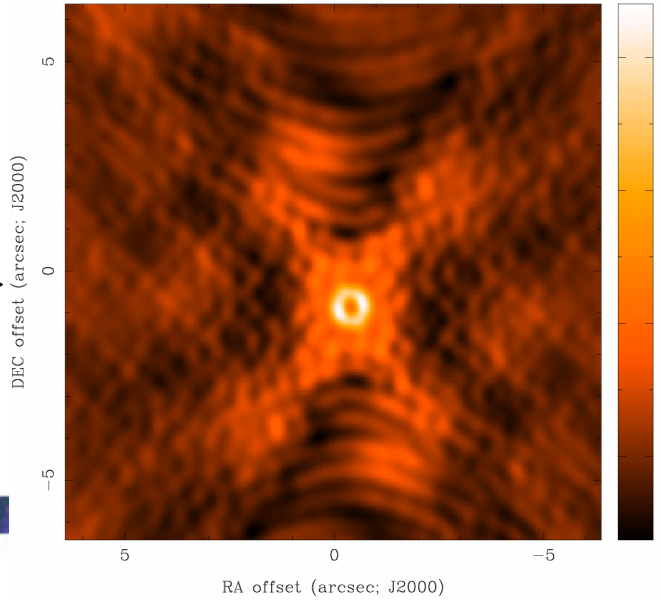




B(u,v)

T(x,y)

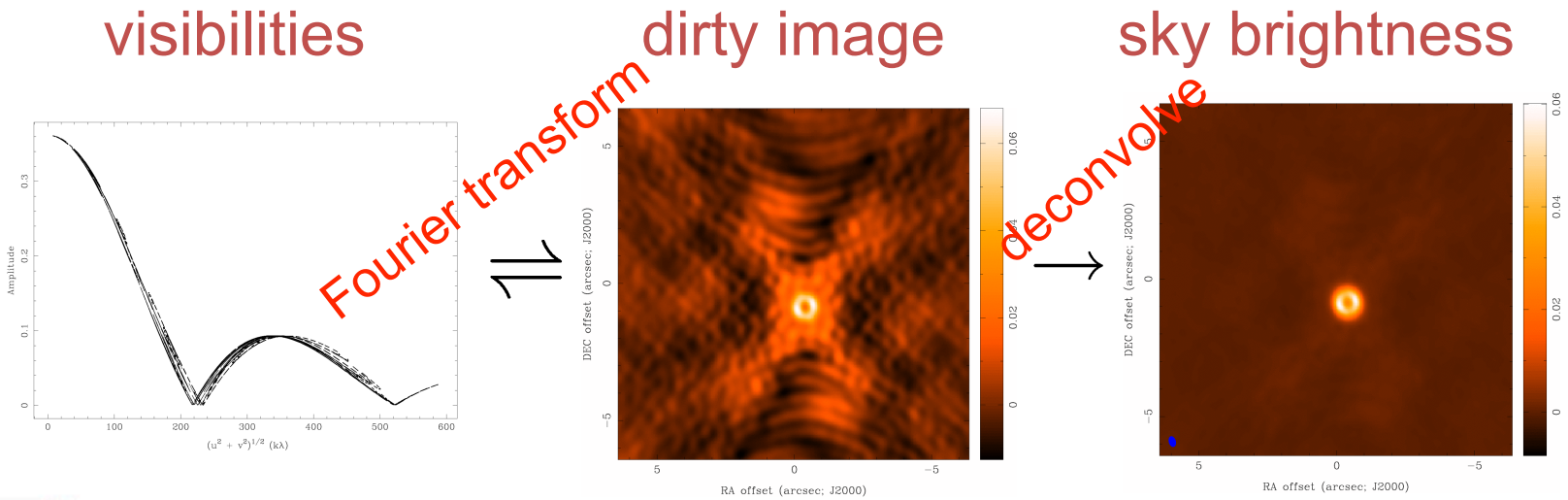




$T^D(x,y)$
(dirty image)

How to analyze (imperfect) interferometer data?

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

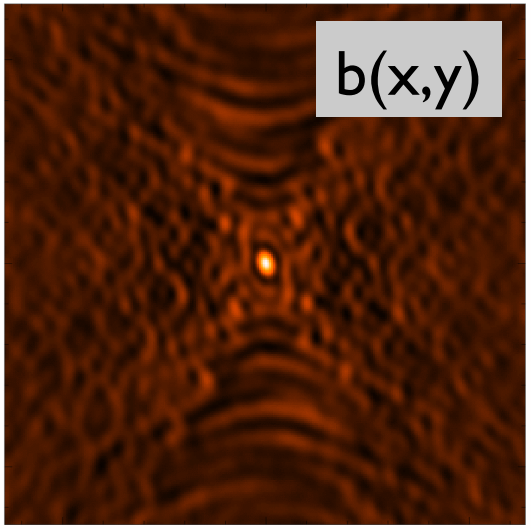


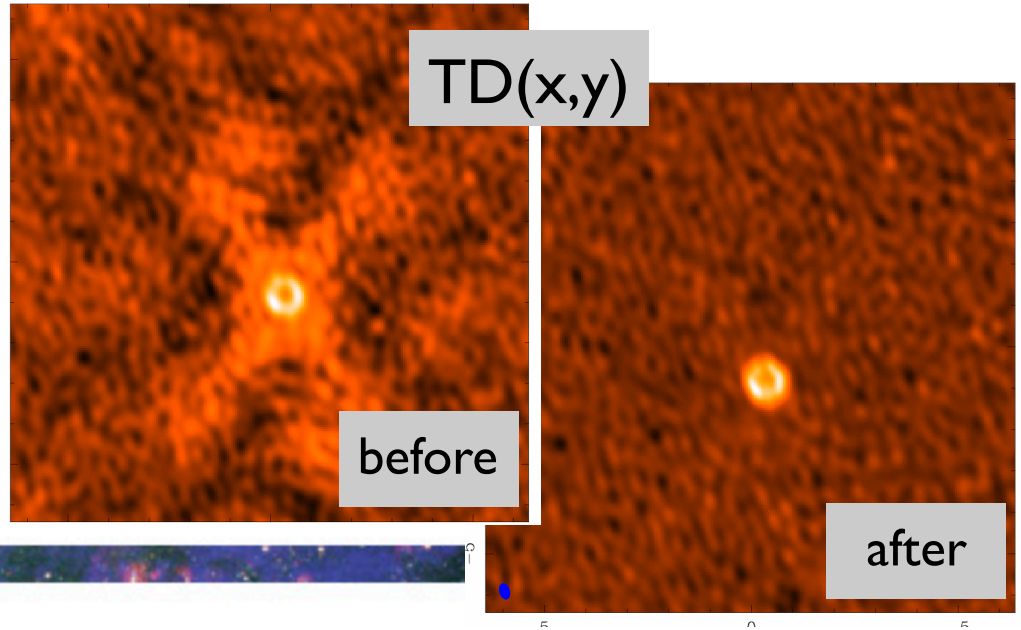
Basic CLEAN Algorithm

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





NRAO

Basic CLEAN Algorithm (cont)

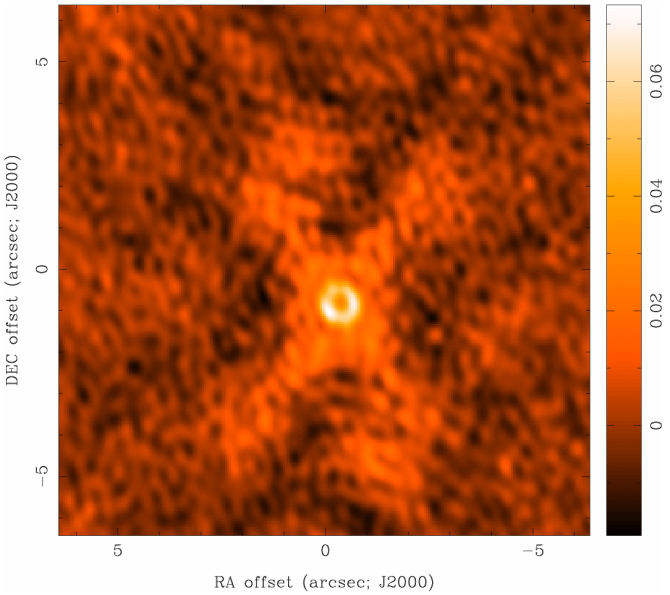
- stopping criteria
  - r*esidual* map max < multiple of rms (when noise limited)
  - r*esidual* map max < fraction of dirty map max (dynamic range limited)
  - max number of clean components reached (no justification)

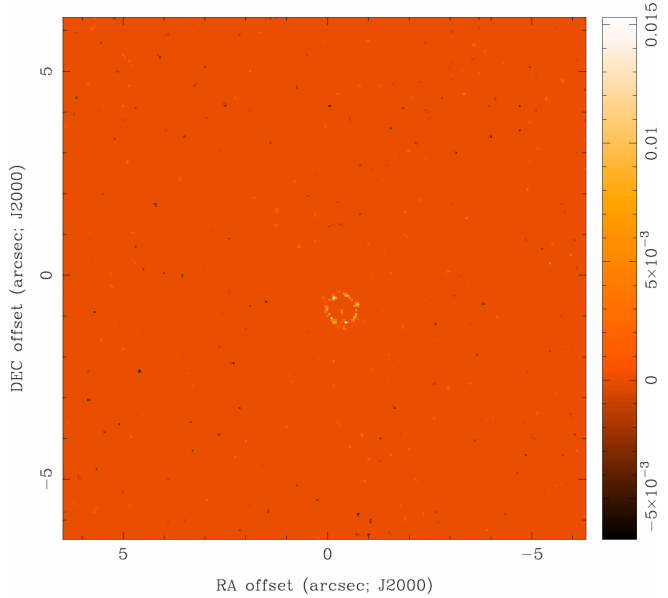
- loop gain
  - good results for g ~ 0.1 to 0.3
  - lower values can work better for smoother emission, g ~ 0.05

- Clean boxes
  - limit the area over which the algorithm searches for clean components
  - Residuals are added back to the image in the end, so “uncleaned” features are not removed.

CLEAN

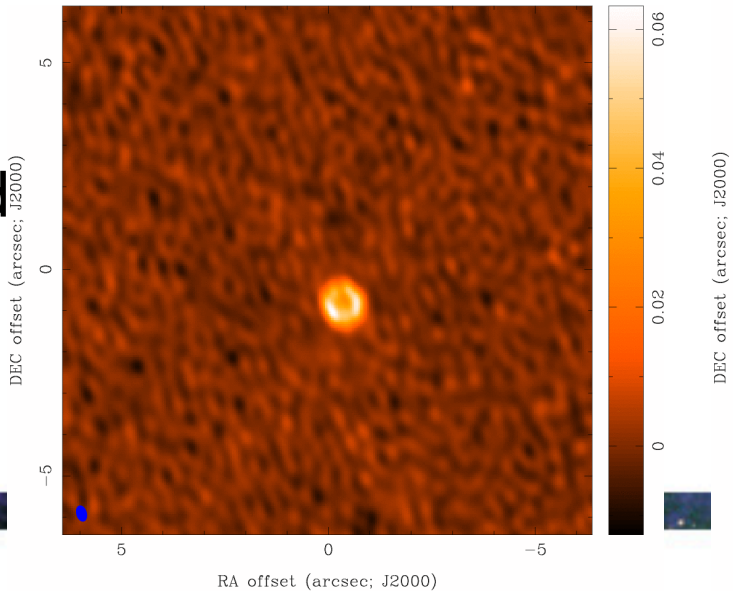
$T^D(x,y)$

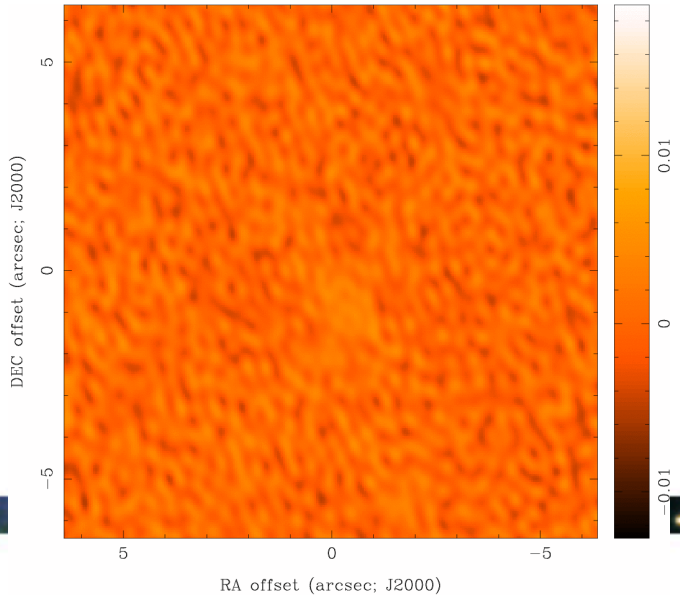




CLEAN model

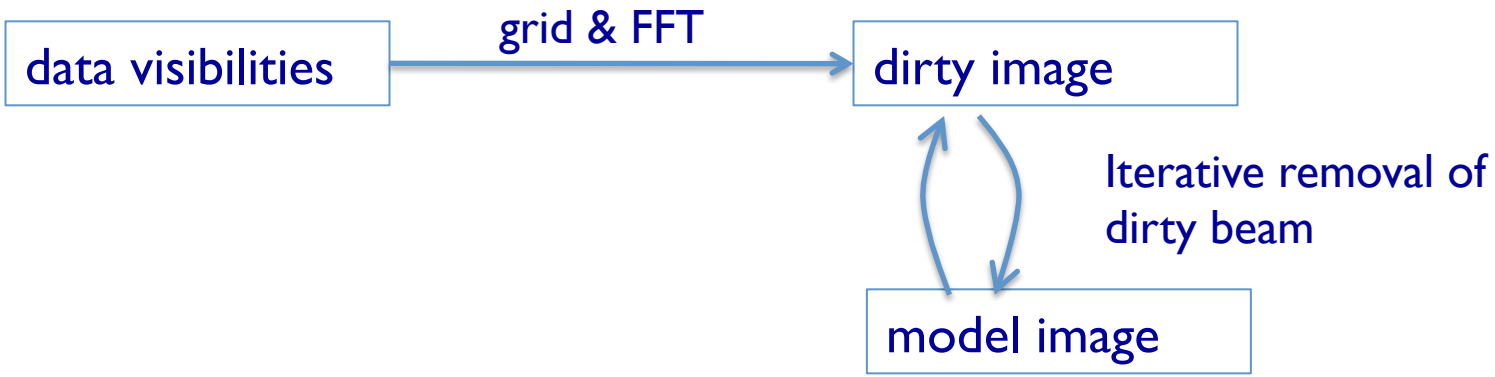
restored image





residual map

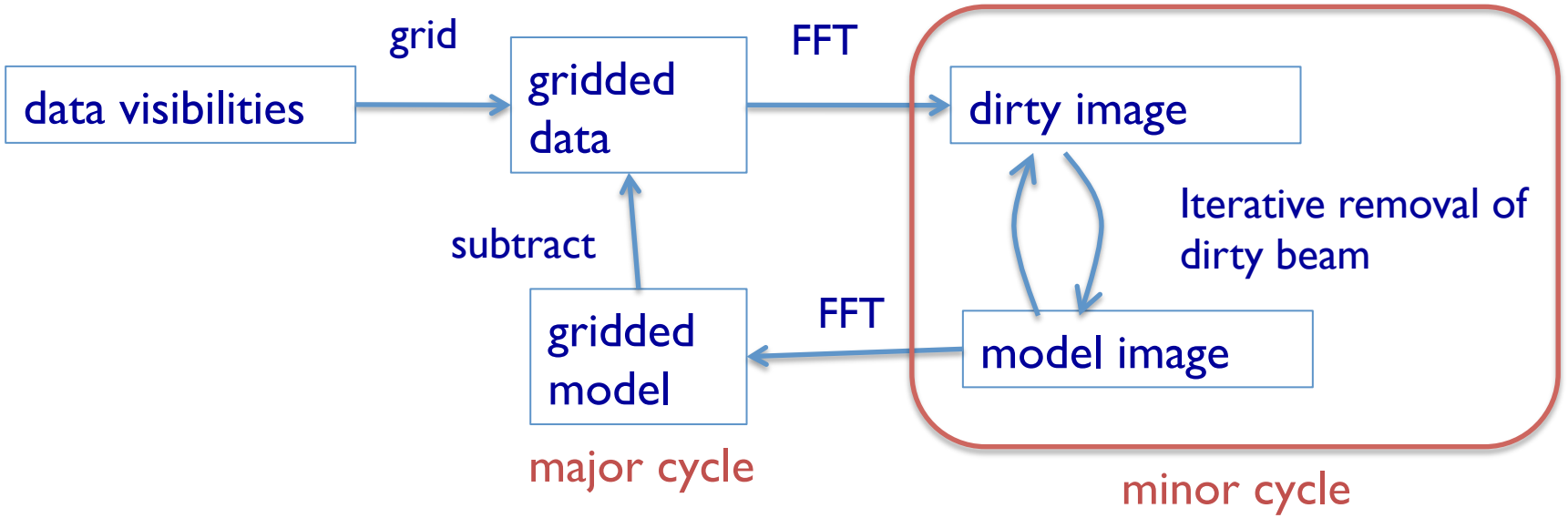
Deconvolution algorithms : Hogbom



subtracts full PSF in image domain

For complex images, errors can build

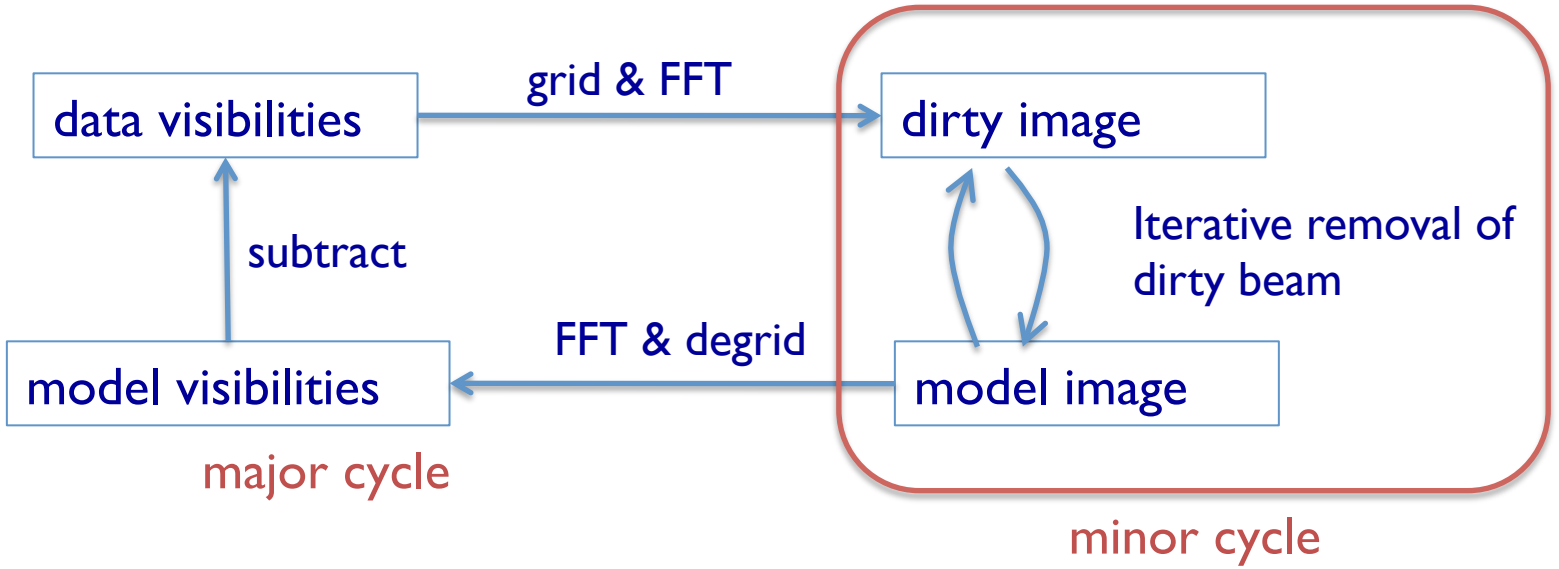
Deconvolution algorithms : Clark



subtracts truncated PSF in image domain

periodically subtracts from gridded data in uv domain

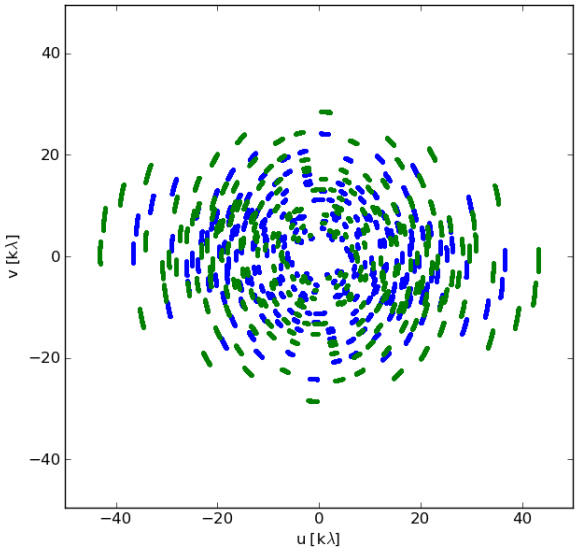
Deconvolution algorithms: Cotton-Schwab



Cotton-Schwab (csclean):

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

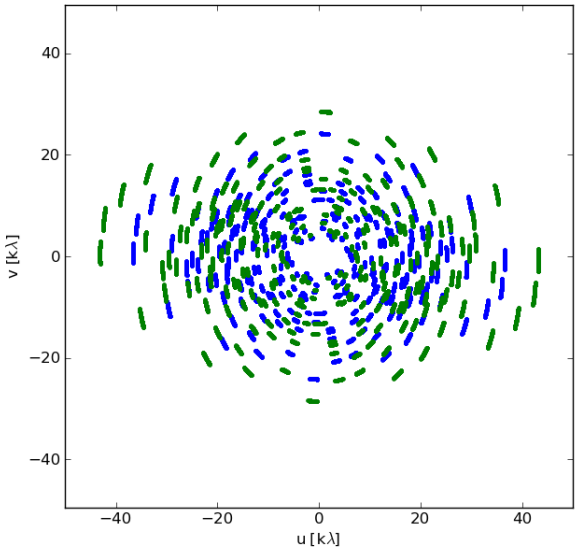
Dirty Beam Shape and Weighting



- Each visibility point is initially weighted by Tsys, integration time

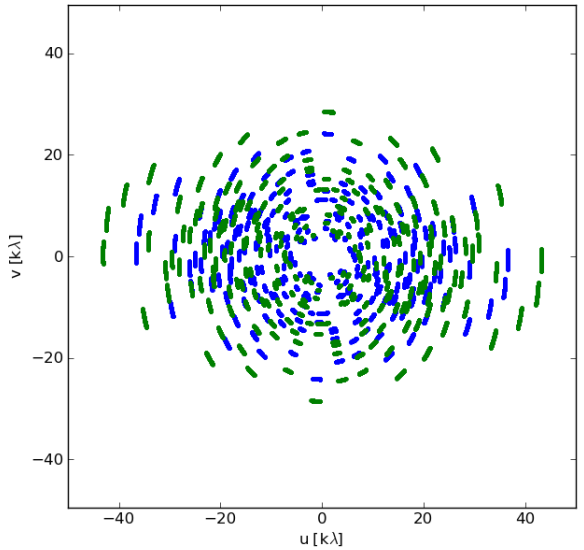
- **Natural**
  - Each sample is given the same weight
  - There are many samples at short baselines, so natural weighting will give the largest beam and the best sensitivity but poor beam characteristics with pronounced wings
- **Uniform**
  - each visibility is given a weight inversely proportional 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 often a good choice.

Dirty Beam Shape and Weighting



- **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



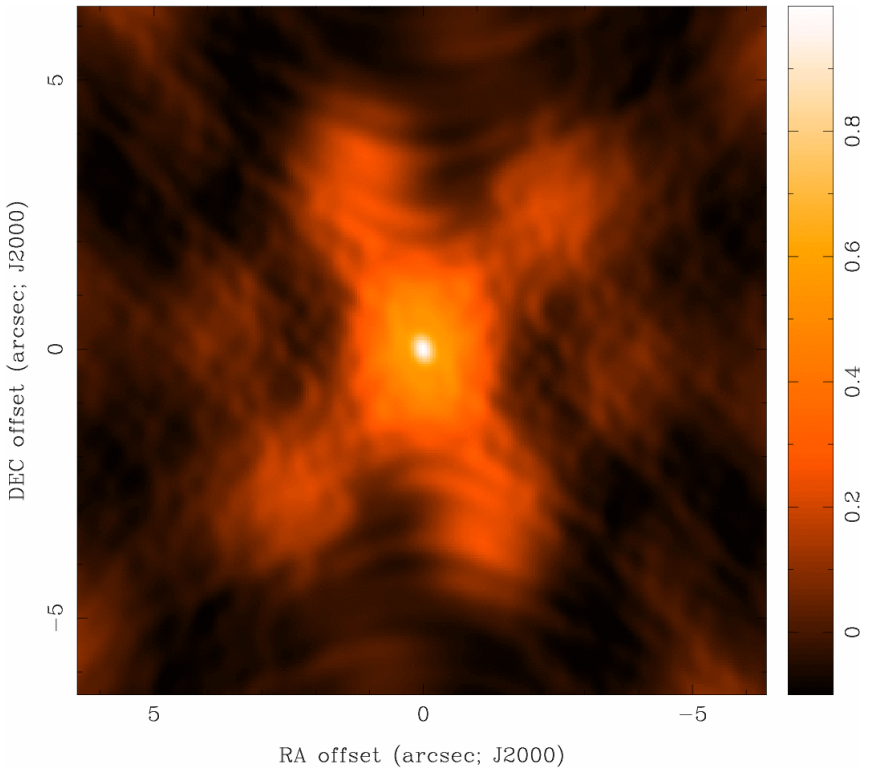
Adjust the weighting to match your science goal:

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

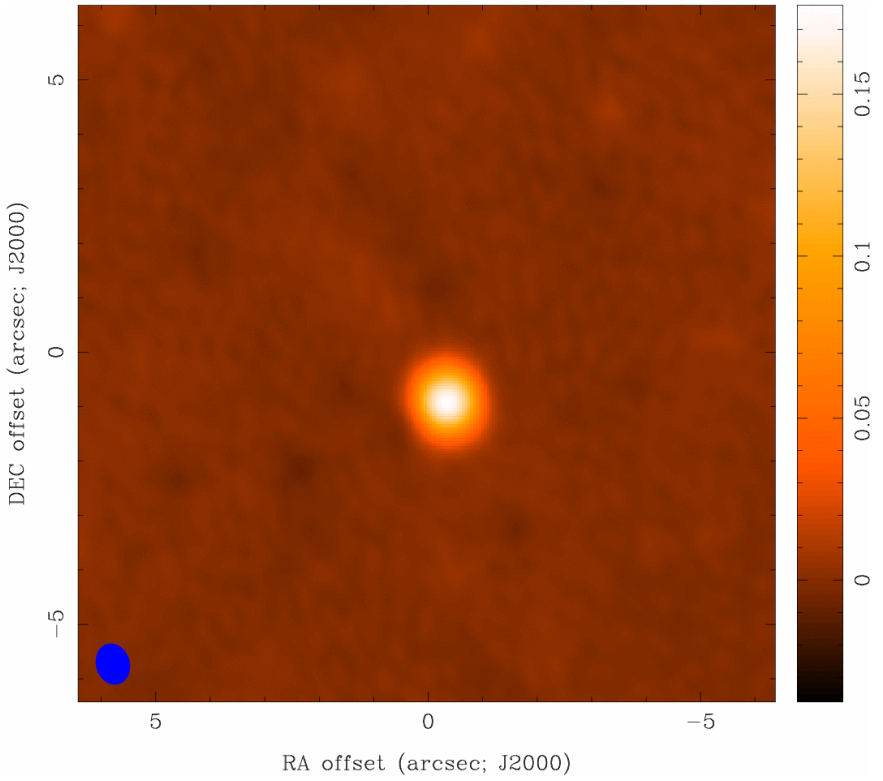
→ Finer detail of strong sources: **robust** or even **uniform**

Imaging Results

Natural Weight Beam

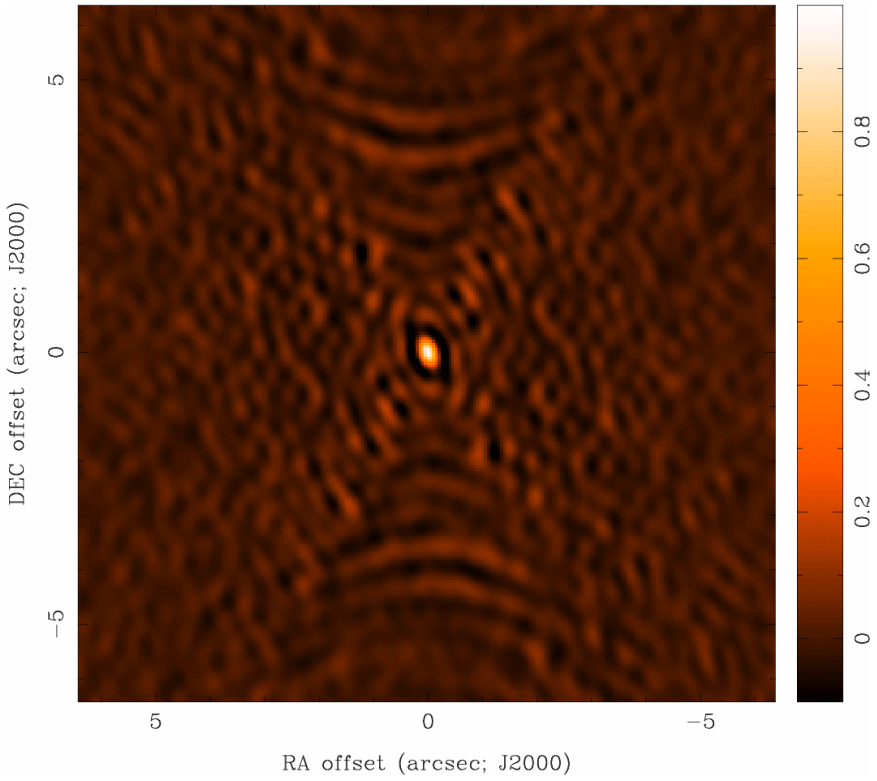


CLEAN image

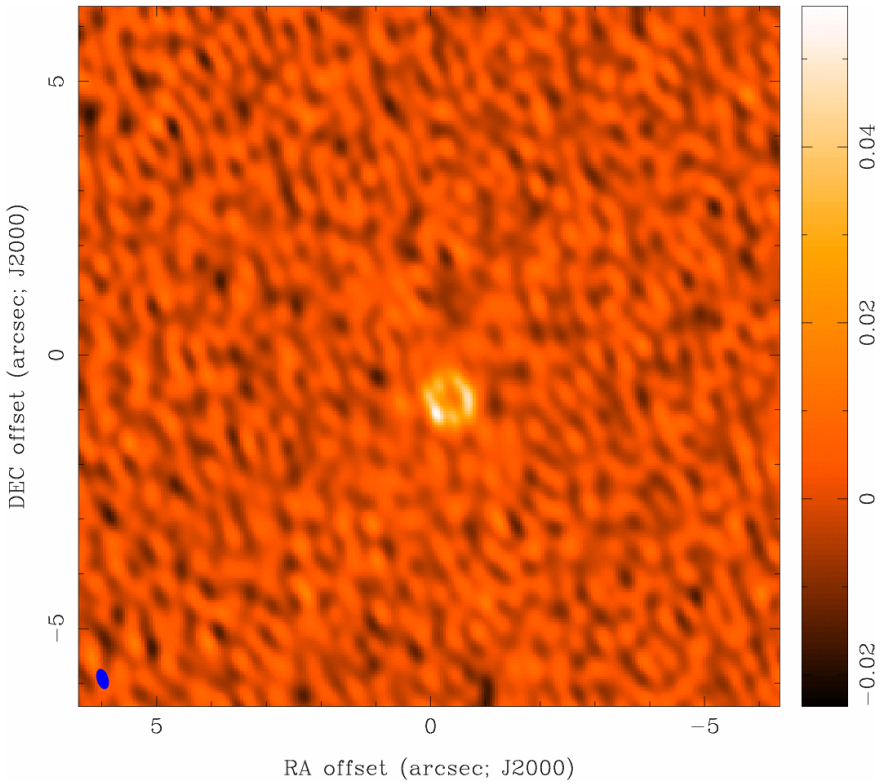


Imaging Results

Uniform Weight Beam

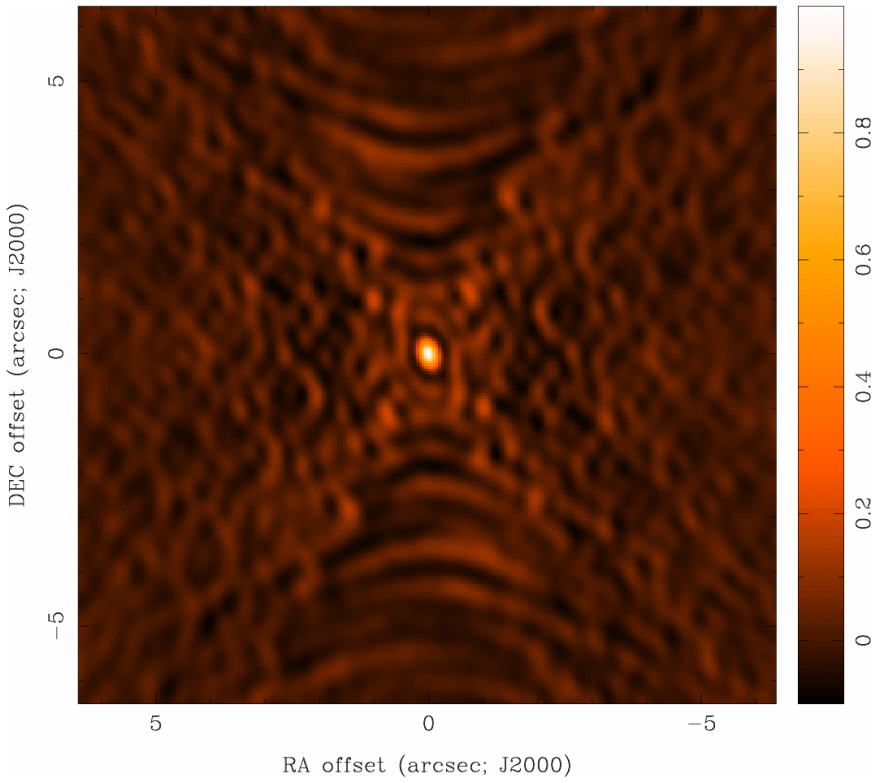


CLEAN image

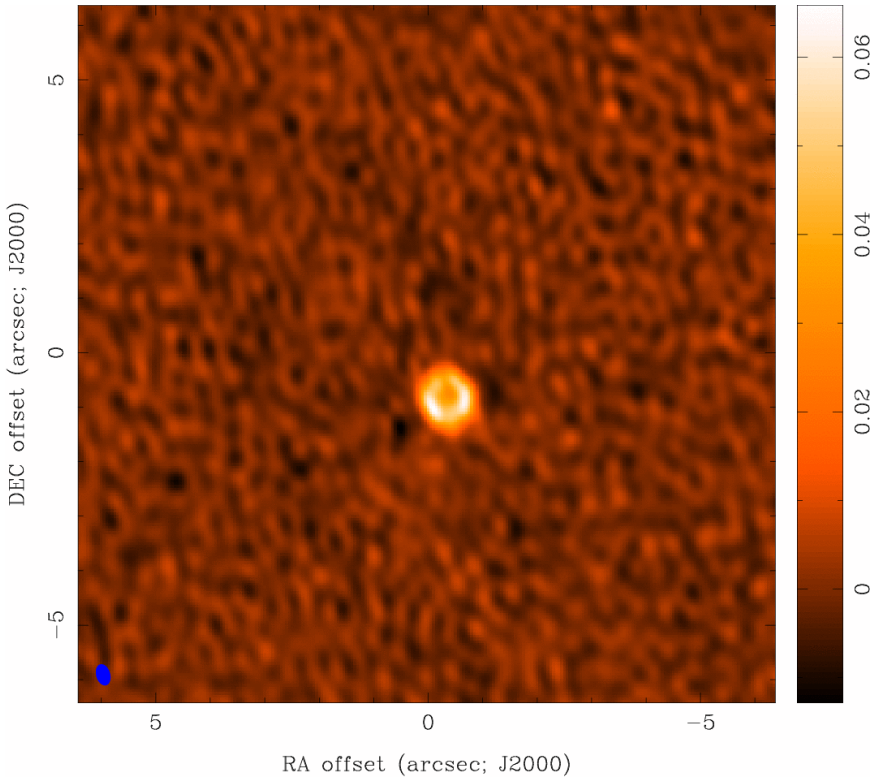


Imaging Results

Robust=0 Beam



CLEAN image



CLEAN in CASA

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

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

Modes/Capabilities:

- continuum: incl. multi-frequency synthesis (radially extends visibilities 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
- multiscale cleaning
- primary beam correction

Clean in CASA:

```
CASA <13>: 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          =      False            #  Other data selection parameters
mode                =      'mfs'            #  Spectral gridding type (mfs, channel, velocity, frequency)
     nterms         =          1            #  Number of Taylor coefficients to model the sky frequency dependence
     reffreq        =         ''            #  Reference frequency (nterms > 1),'' uses central data-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
async               =      False            #  If true the taskname must be started using clean(...)
```

Basic Image Parameters: Pixel Size and Image Size

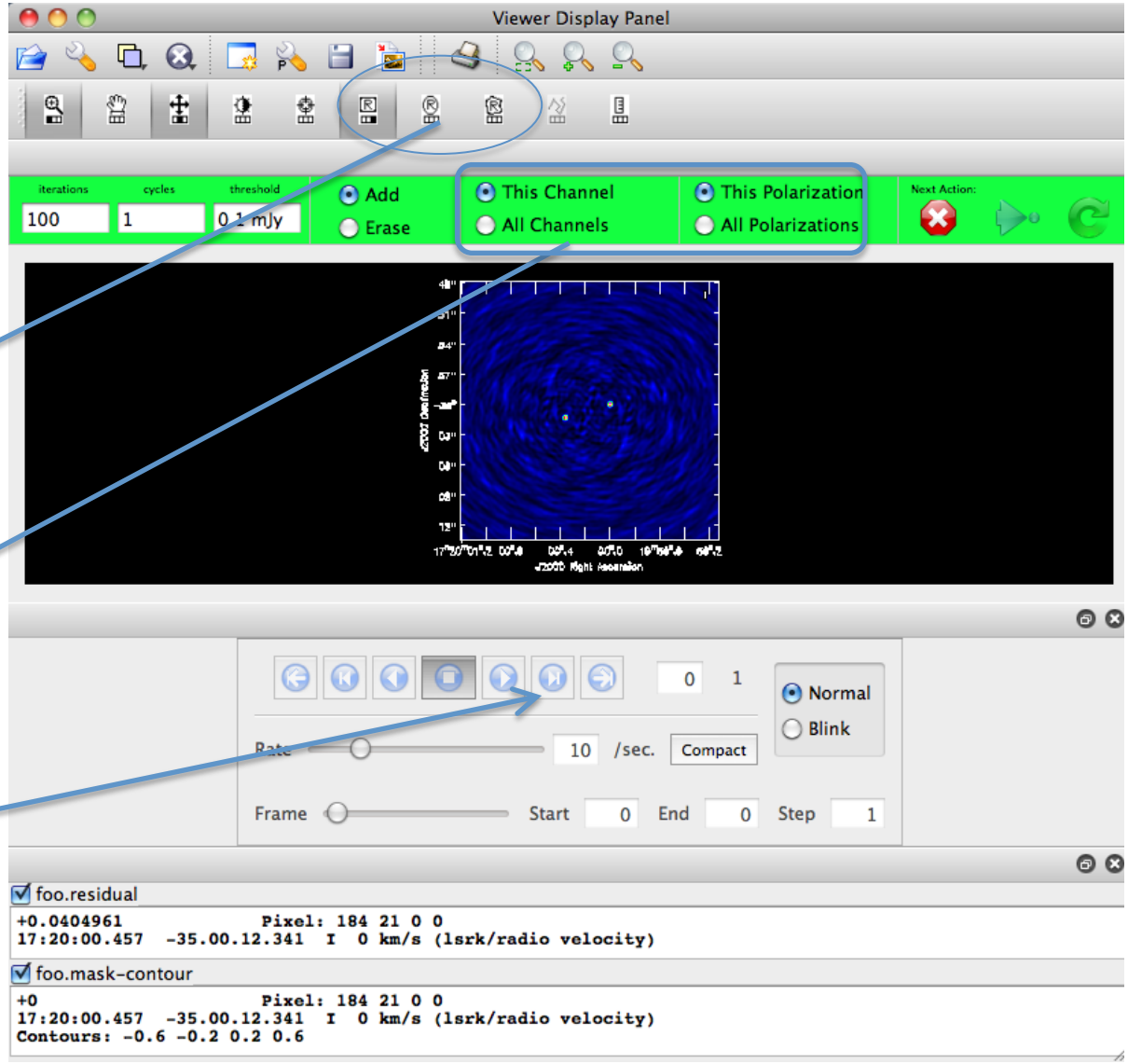
- pixel size
  - should satisfy $\Delta x < 1/(2\, u_{max})$ $\Delta y < 1/(2\, v_{max})$
  - in practice, 3 to 5 pixels across the main lobe of the dirty beam

- image size
  - Consider FWHM of primary beam (e.g. ~ 20" at Band 7)
  - Be aware that sensitivity is not uniform across the primary beam
  - Use mosaicing to image larger targets
  - Not restricted to powers of 2

  * if there are bright sources in the sidelobes, they will be aliased into the image (need to make a larger image)

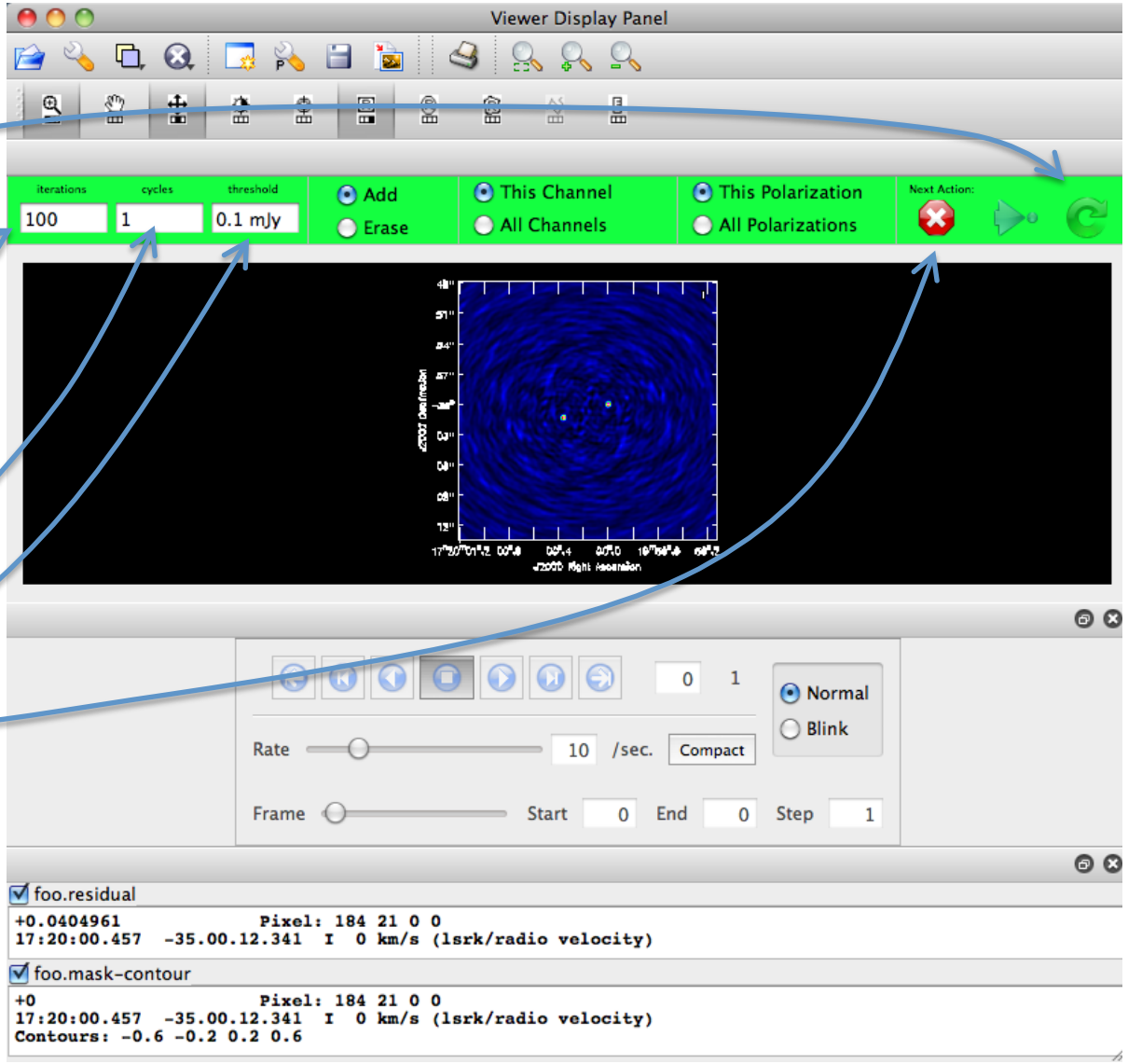
Interactive Clean

- residual image in viewer
- define a mask with R-click 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



Output of CLEAN

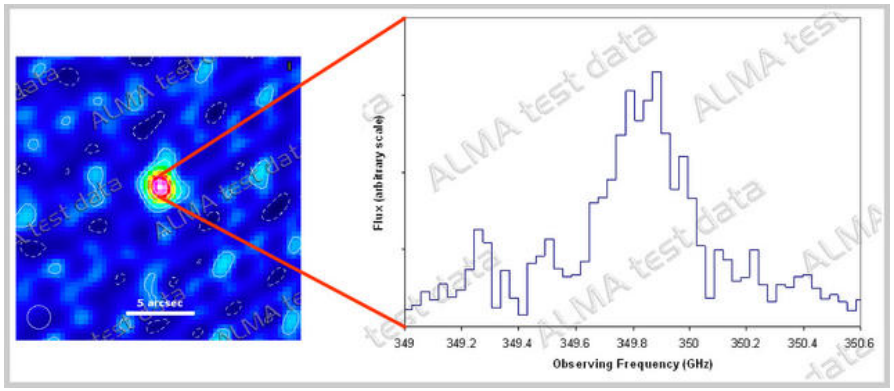
Minimally:

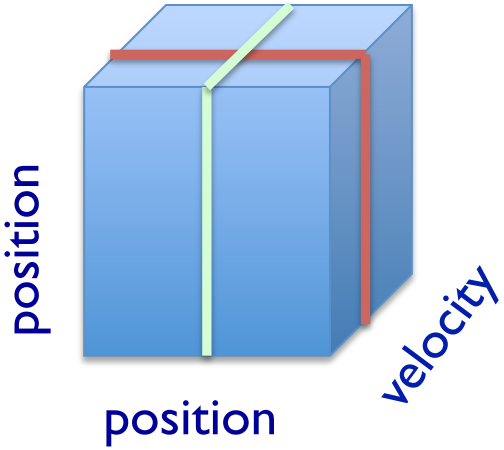
• my_image.flux	Relative sky sensitivity
• my_image.image	Cleaned and restored image (Jy/clean beam)
• my_image.mask	Clean "boxes"
• my_image.model	Clean components (Jy/pixel)
• my_image.psf	Dirty beam
• my_image.residual	Residual (Jy/dirty beam)

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.*')

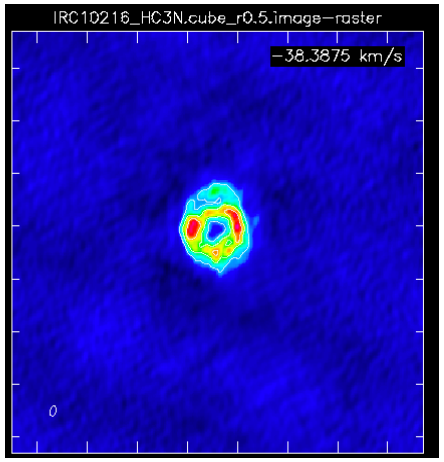
Some of the capabilities of *clean* …

Imaging spectral lines

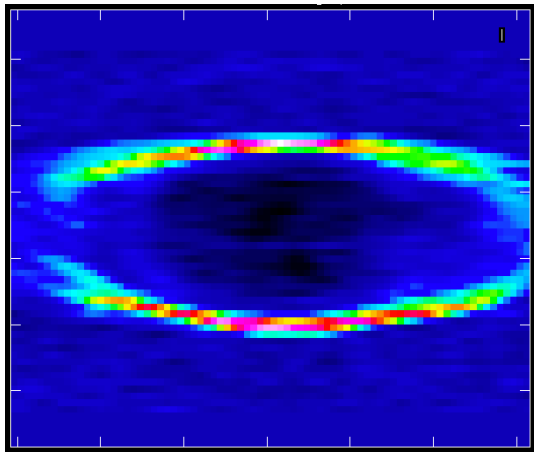




- Channel map



- Position-velocity map



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)
```

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)

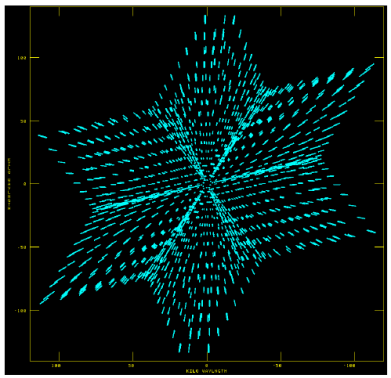
Imaging spectral lines: continuum subtraction

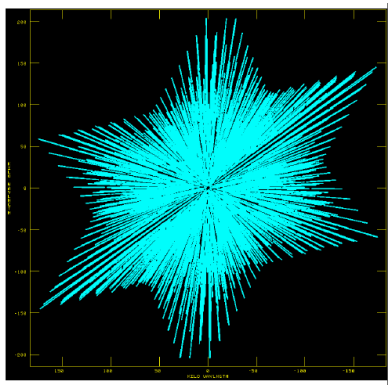
- Generally would like to subtract continuum emission (we will see how to identify line-free channels in hands-on session)
- Use `uvcontsub` to do the subtraction in uv plane.

```
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

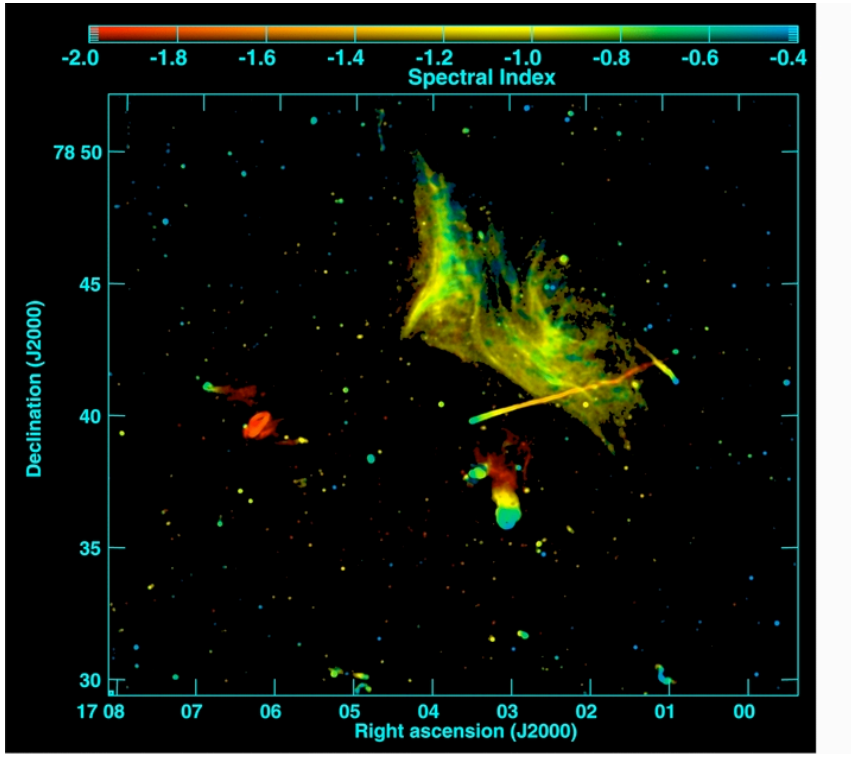
- Multi-scale Multi-Frequency Taylor Term expansion





Narrow BW wide BW

(better uv-coverage)



Abell 2256; Owen et al. (2014)

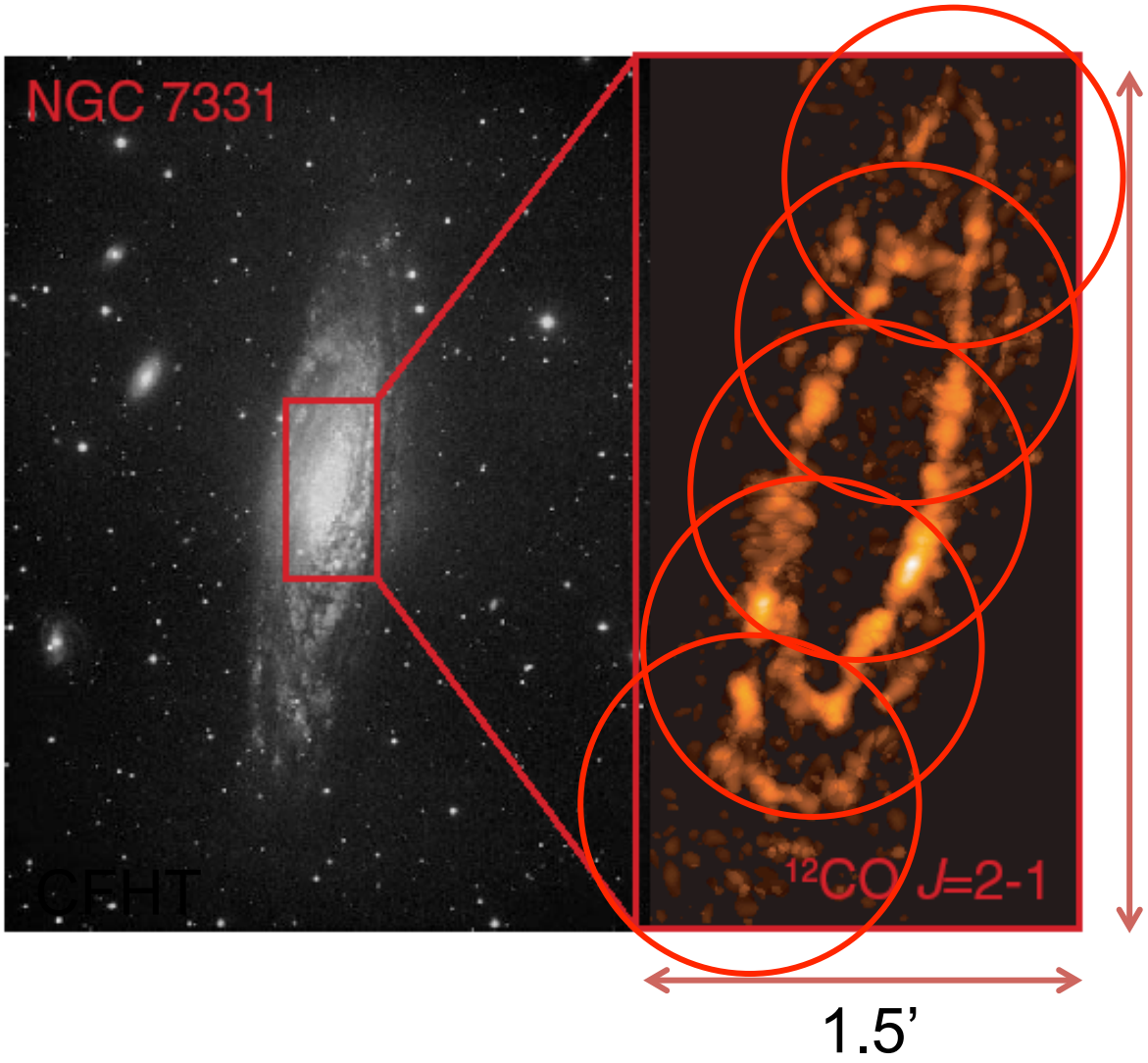
Multi-Frequency Synthesis (mfs)

In clean:

```
mode                =       'mfs'        #  Spectral gridding type (mfs, channel, velocity, frequency)
     nterms         =           1        #  Number of Taylor coefficients to model the sky frequency dependence
     reffreq        =          ''        #  Reference frequency (nterms > 1),'' uses central data-frequency
```

- nterm=2 computes spectral index, 3 for curvature
- needed for bandwidths ~5% or more (S/N dependent)

Mosaics



Example: SMA 1.3 mm observations: 5 pointings

- Primary beam ~1'
- Resolution ~3"

ALMA 1.3mm PB

ALMA 0.85mm PB

The number of pointings is currently limited by policy

Petitpas et al.

Imaging mosaics

```
imagermode          =   'mosaic'
     mosweight      =      False
     ftmachine      =       'ft'
     scaletype      =    'SAULT'

     cyclefactor    =        1.5
     cyclespeedup   =         -1
     flatnoise      =       True
```

```
#  Options: 'csclean' or 'mosaic', '', uses psfmode
#  Individually weight the fields of the mosaic
#  Gridding method for the image
#  Controls scaling of pixels in the image plane. default='SAULT'; example:
#   scaletype='PBCOR' Options: 'PBCOR','SAULT'
#  Controls how often major cycles are done. (e.g. 5 for frequently)
#  Cycle threshold doubles in this number of iterations
#  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)
```

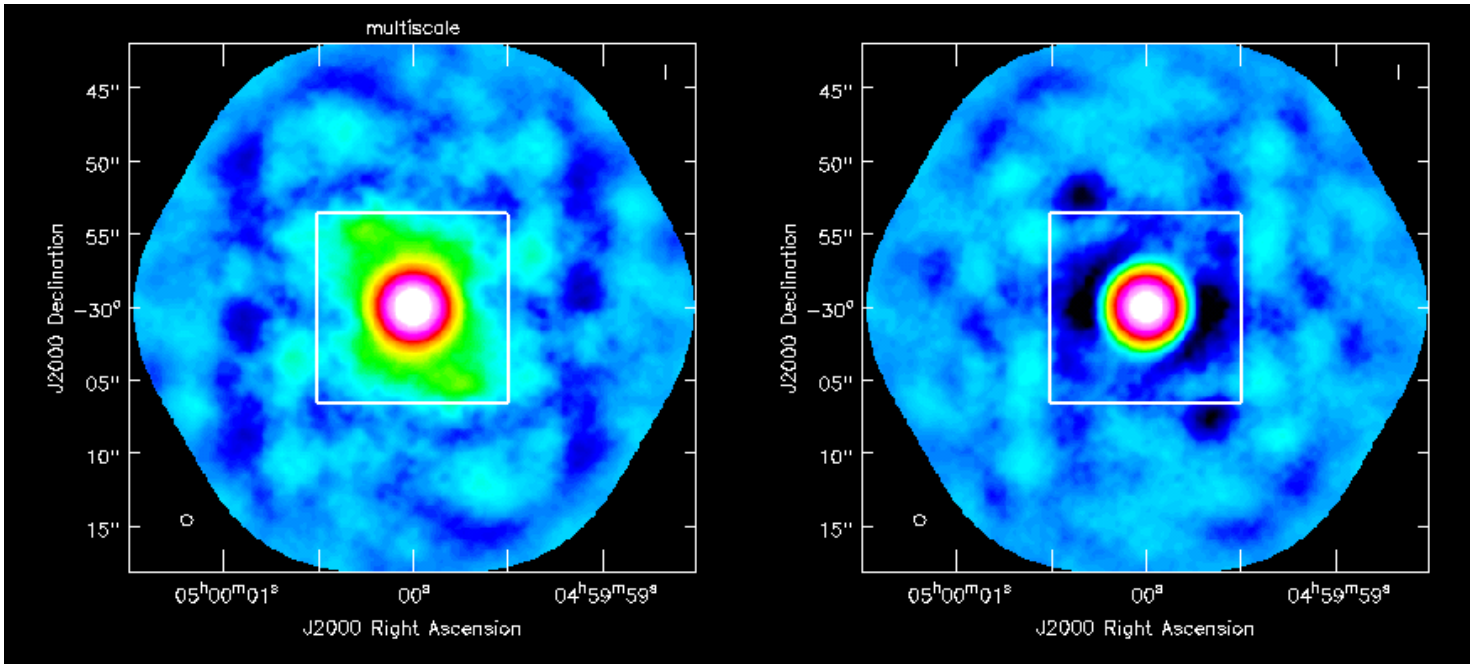
ftmachine = "ft" : shift and add in image plane

ftmachine = "mosaic" : add in uv plane and invert together

Multi-scale CLEAN

multi-scale

"classic" scale



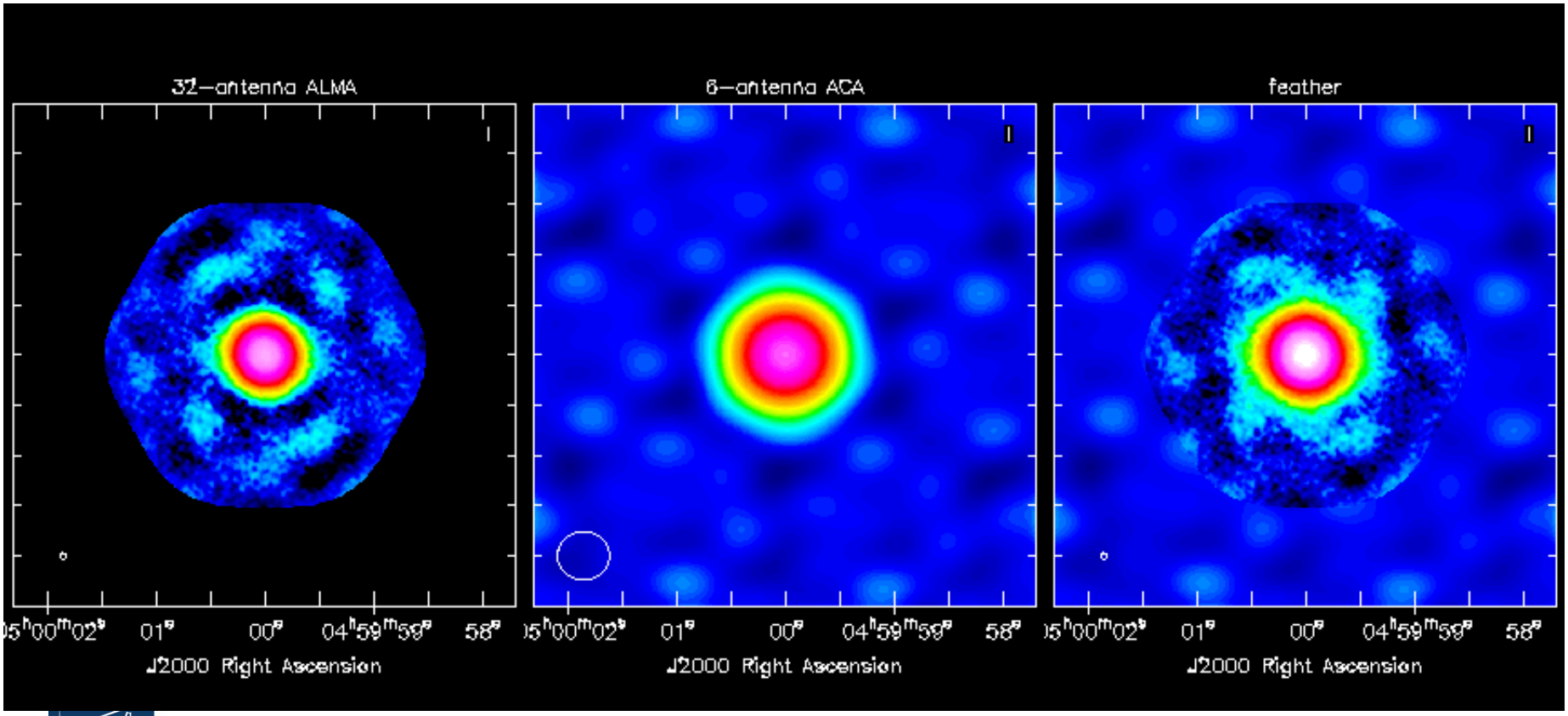

```
multiscale          = [0, 5, 12, 24, 50] #  Deconvolution scales (pixels); [] =
                                         #  standard clean
```

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

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(...)
```



A few more notes …

- The clean mask in can be reused but be careful of imsize.
- The total cleaned flux is not reported until the end.
- *Don't do ^C while imaging – this can do bad things to your MS!*

The End