From calibrated visibilities to images - 1

$$V^{cal} = S \cdot V^{true}$$

S is the sampling function:
S(u,v) = 1 at (u,v) points where visibilities are measured;
S(u,v) = 0 elsewhere

The measured brightness is:

$$I^{meas} = FT^{-1}\left(V^{cal}\right) = FT^{-1}(S) * FT^{-1}(V^{true})$$

called also **Dirty Image**

From calibrated visibilities to images - 2

$$I^{meas} = FT^{-1}\left(V^{cal} \right) = FT^{-1}(S) * FT^{-1}(V^{true})$$

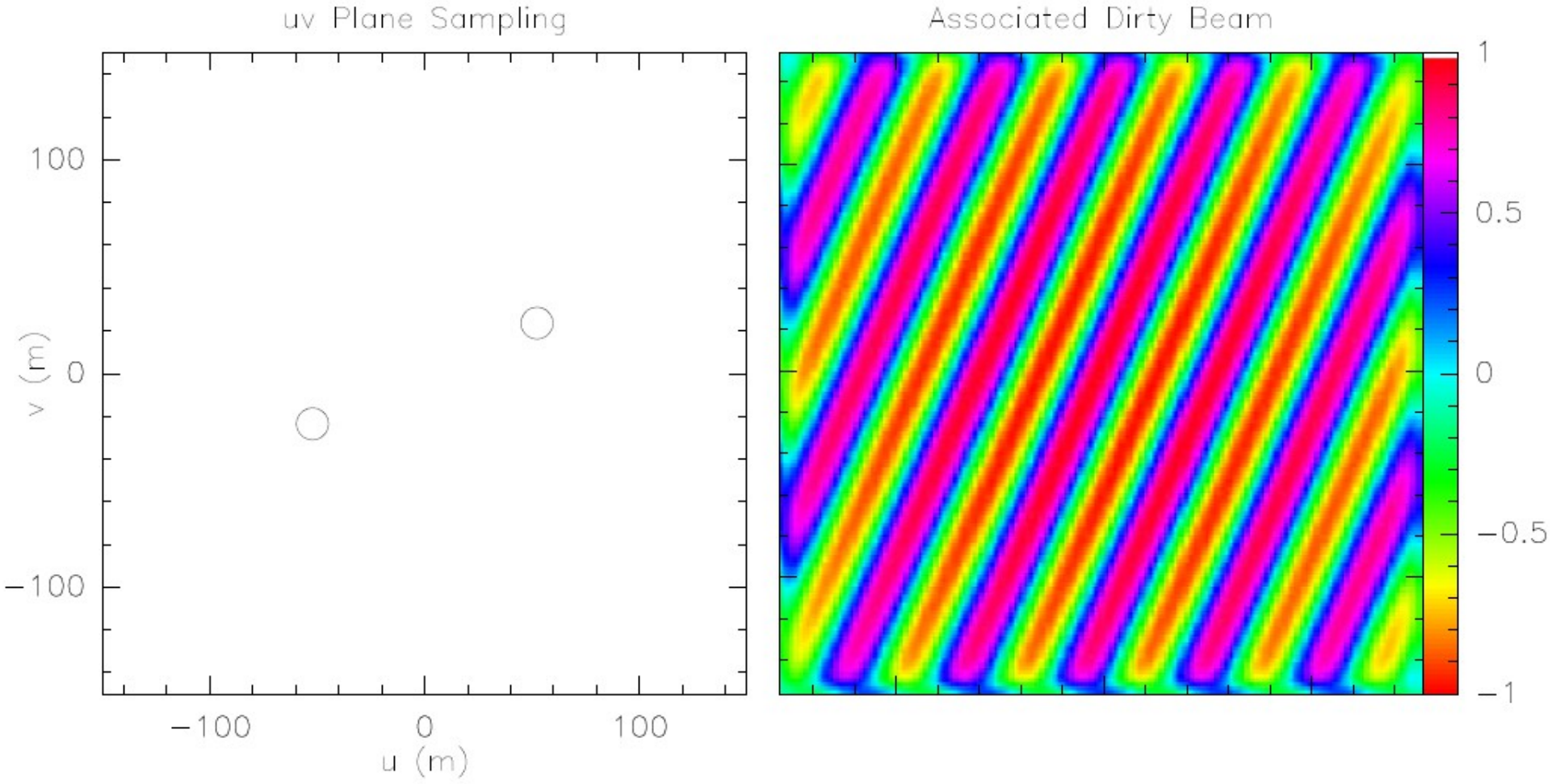
The FT^{-1} of sampled visibilities is the true sky brightness convolved with the PSF:

$$B^{dirty} = FT^{-1}(S)$$ **Dirty beam**

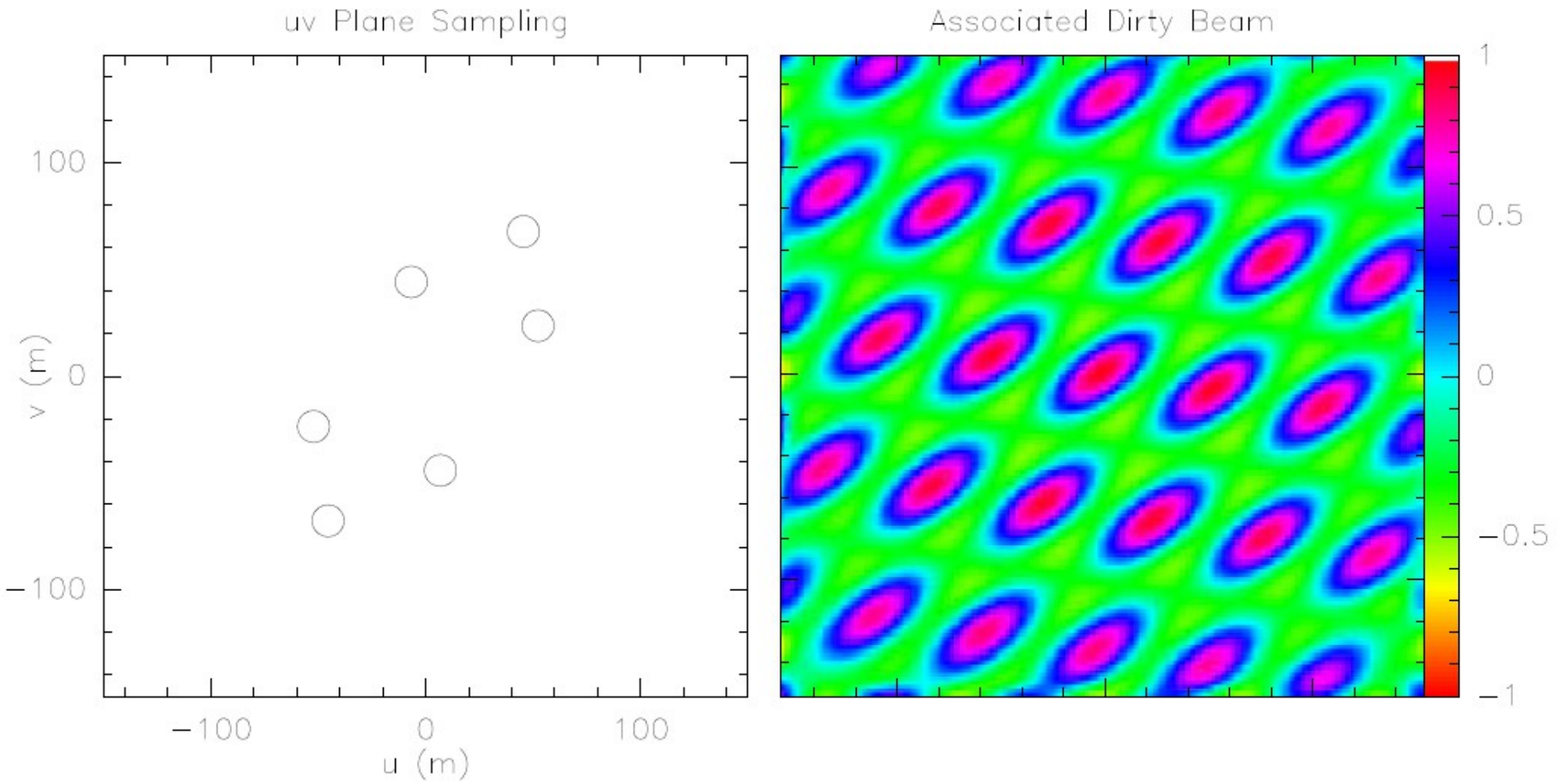
For a perfect sampling function (S=1 everywhere) the dirty beam is a Dirac distribution.

In real world the closest the dirty beam is to a **gaussian** the better.

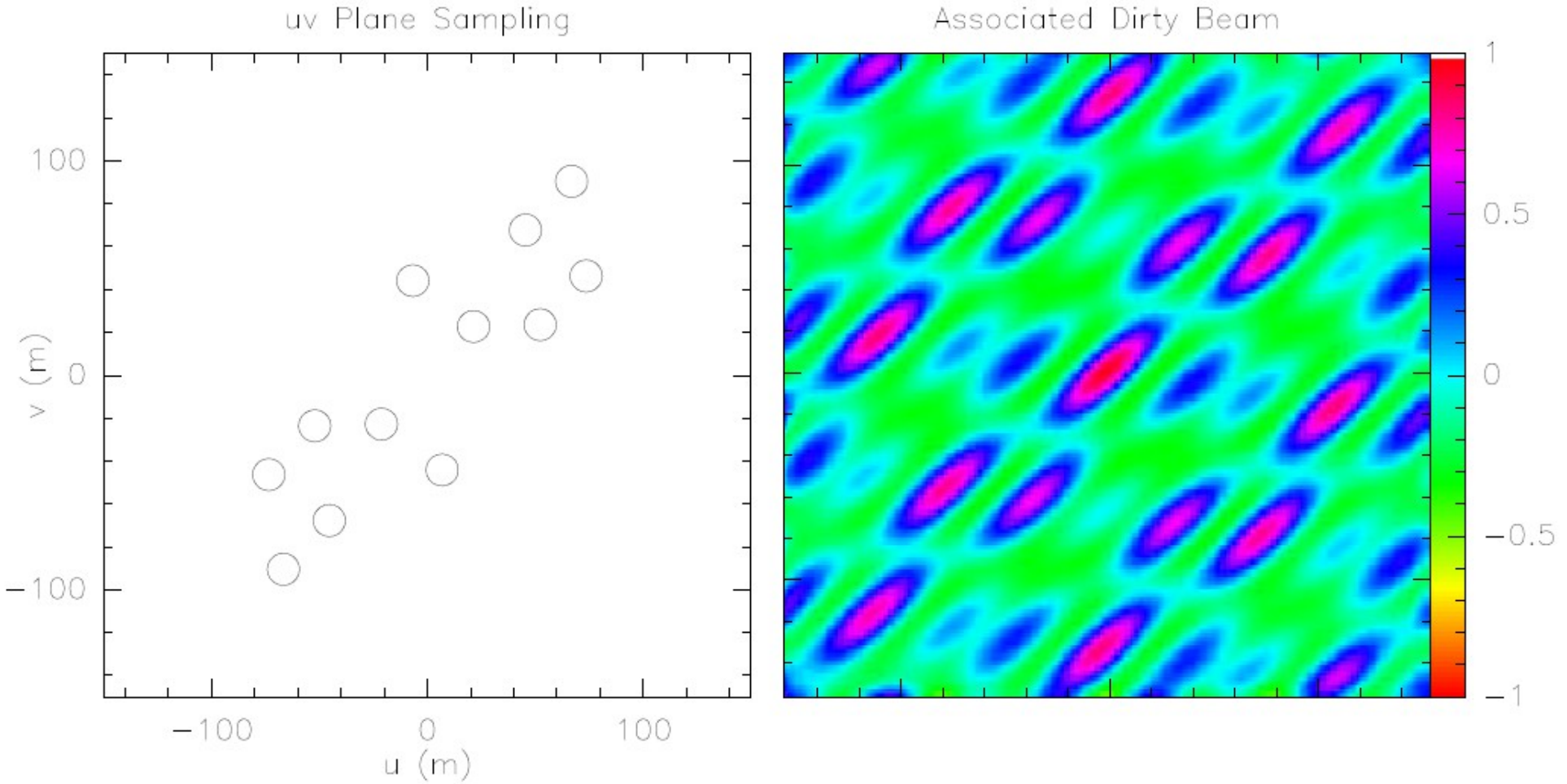
Dirty Beam Shape and Number of Antenna:
2 Antenna



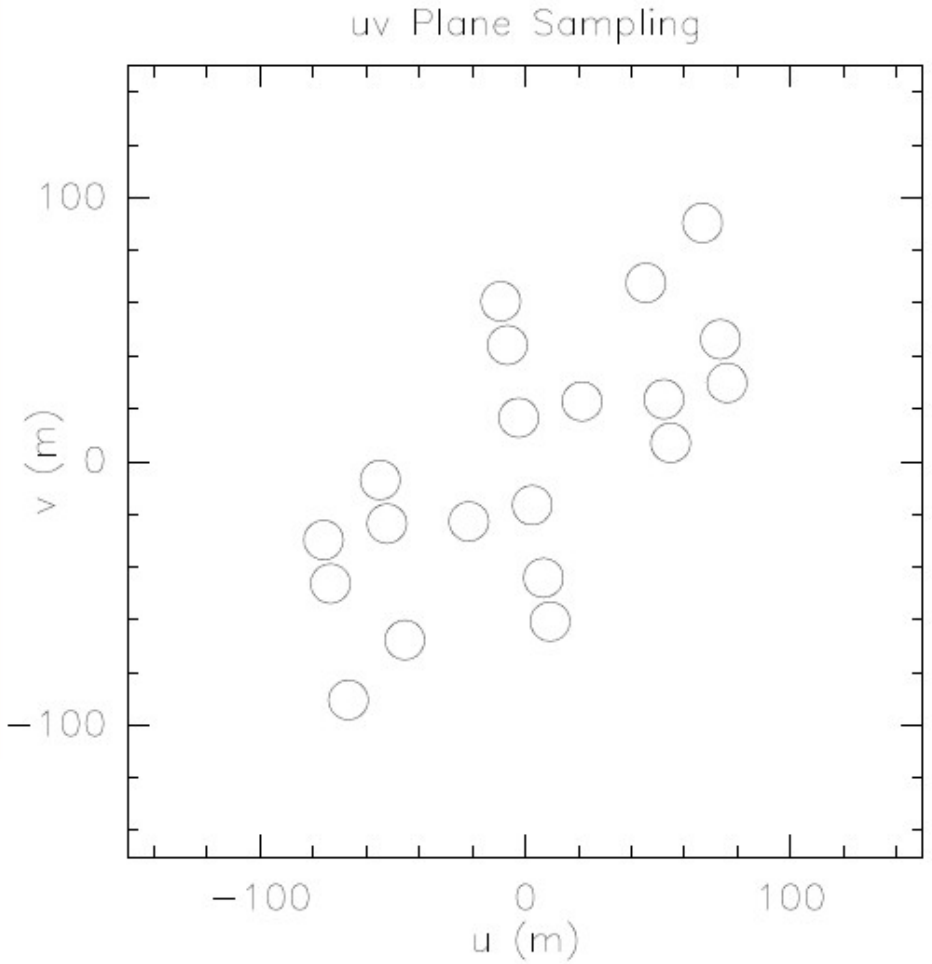
Dirty Beam Shape and Number of Antenna:
3 Antenna

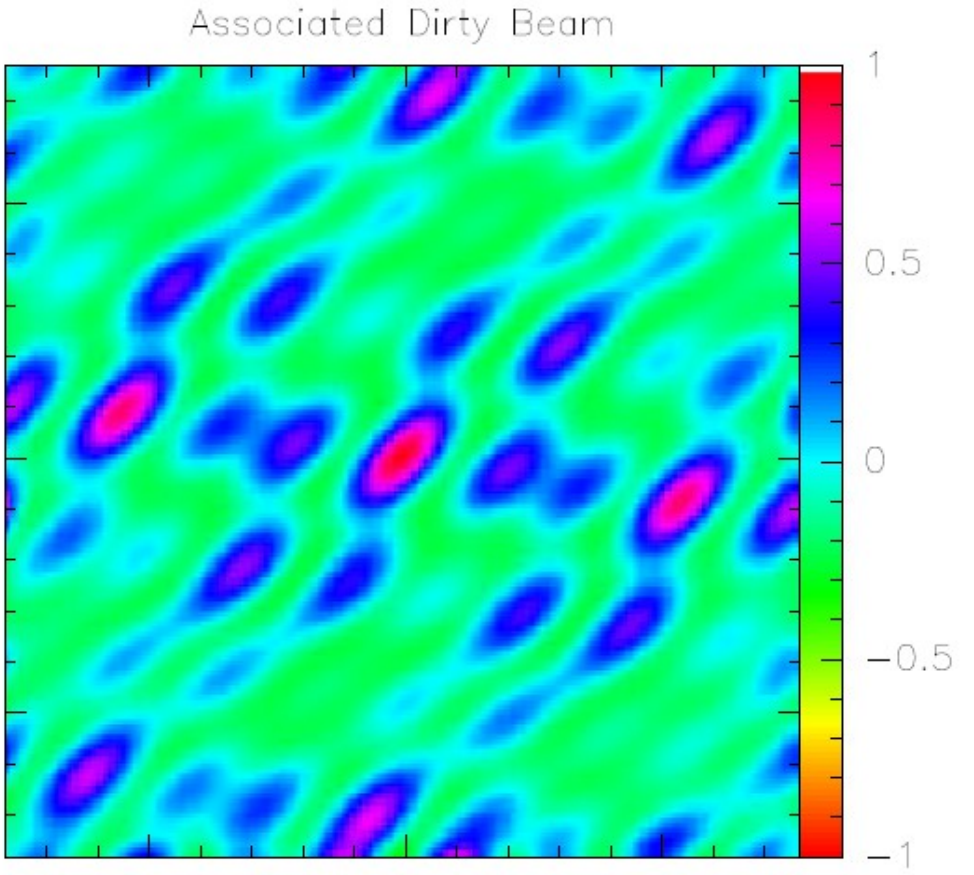


Dirty Beam Shape and Number of Antenna:
4 Antenna

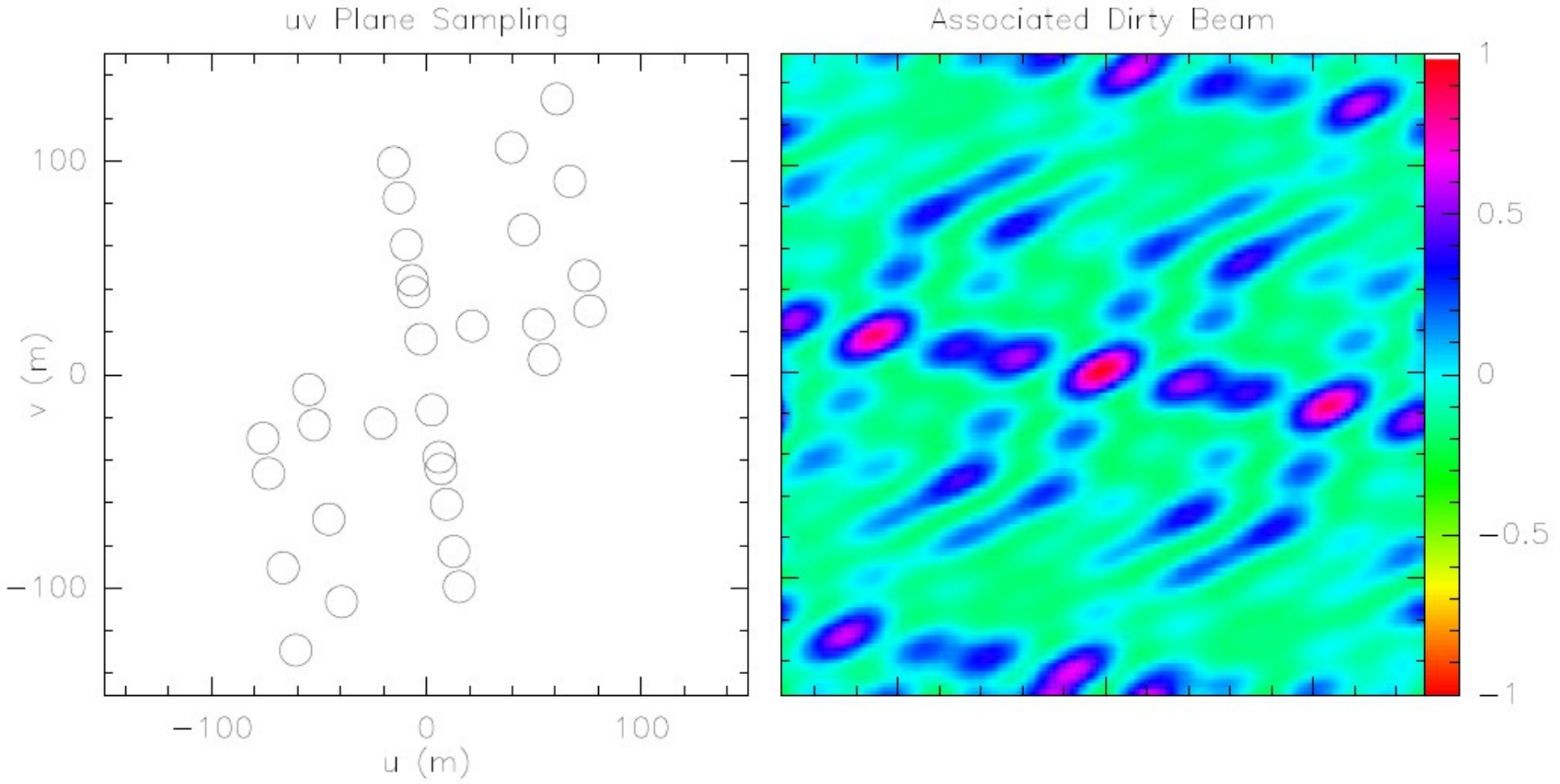


Dirty Beam Shape and Number of Antenna:
5 Antenna

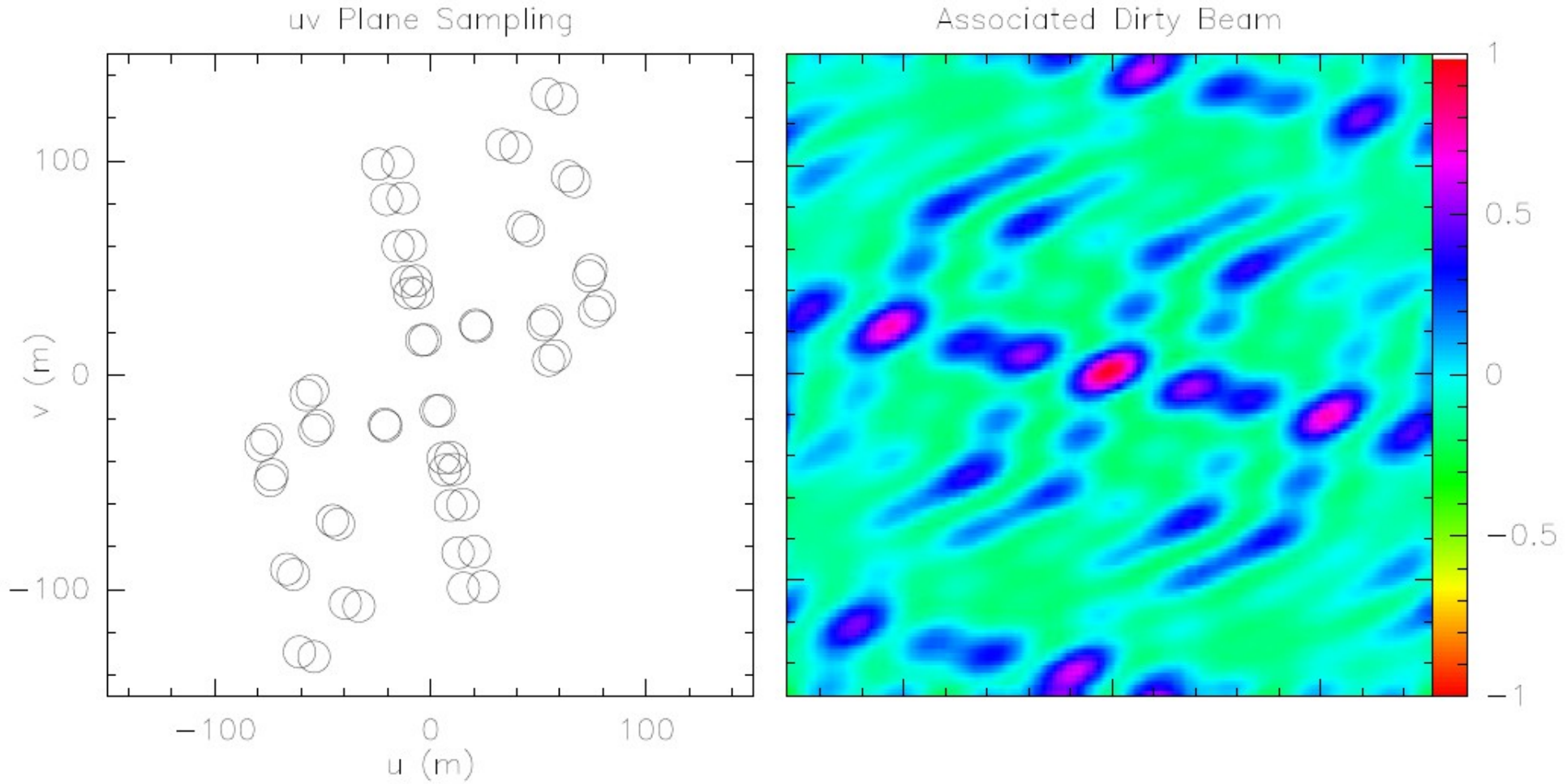




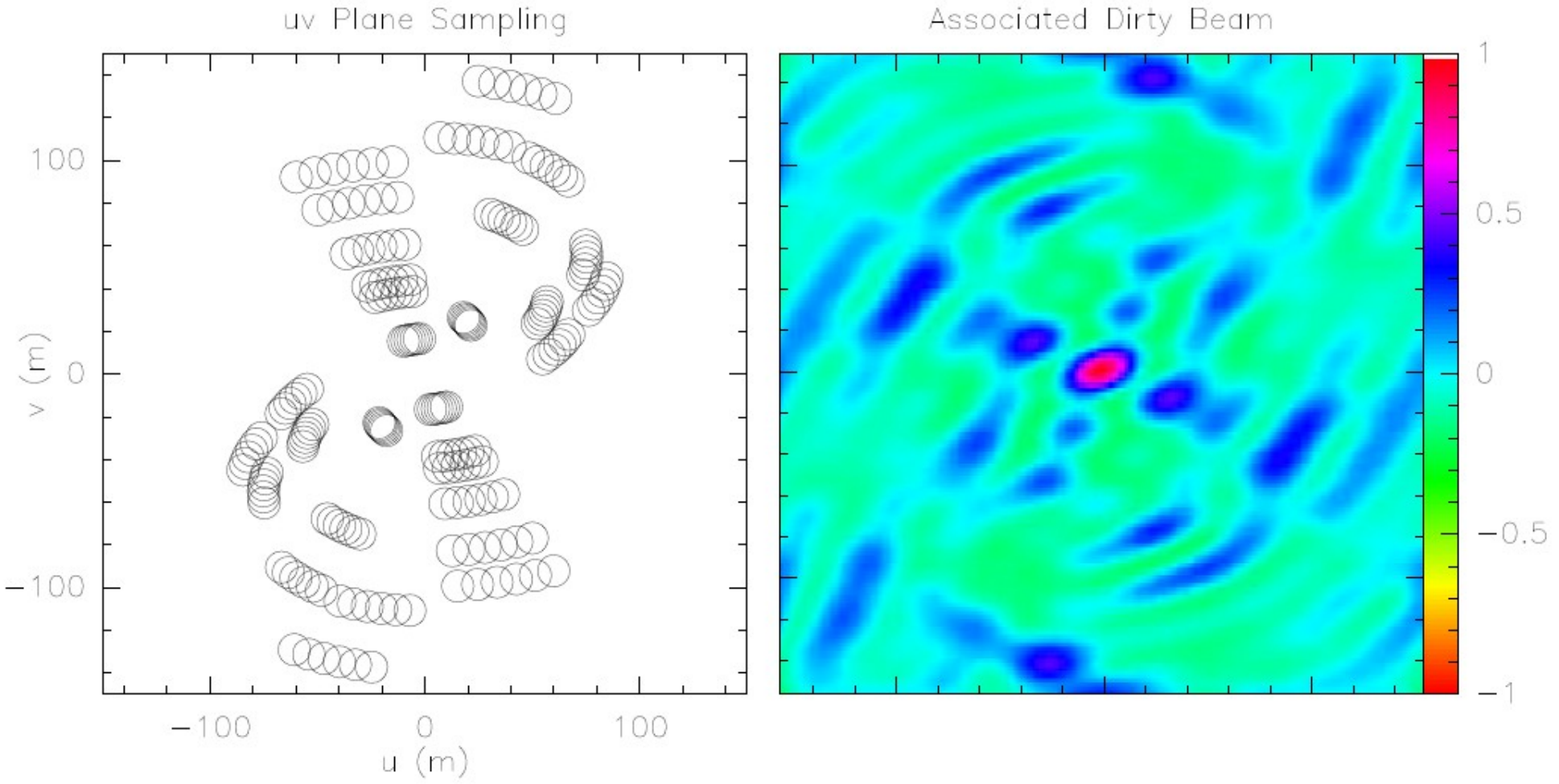
Dirty Beam Shape and Number of Antenna:
6 Antenna



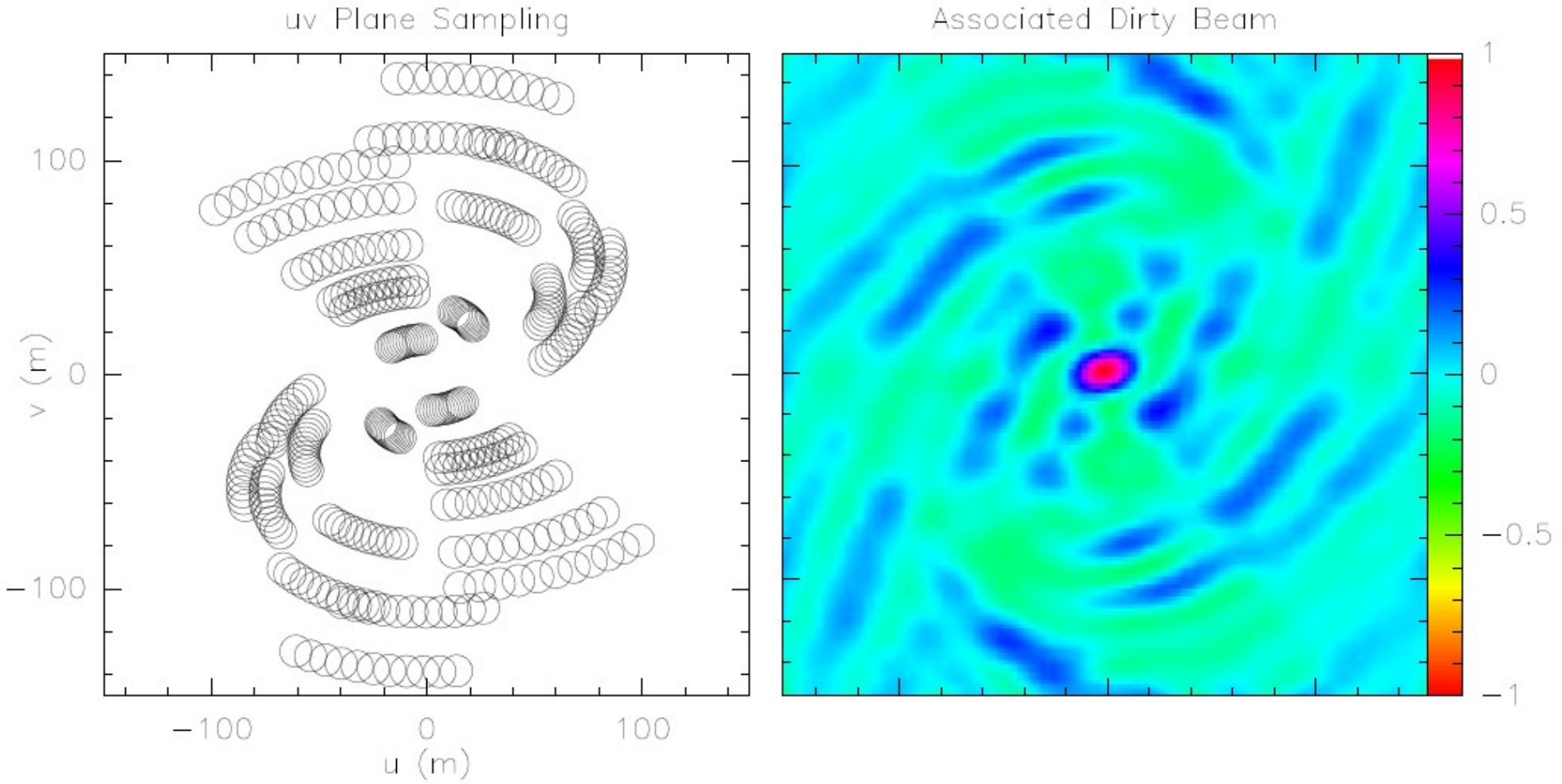
Dirty Beam Shape and Super Synthesis



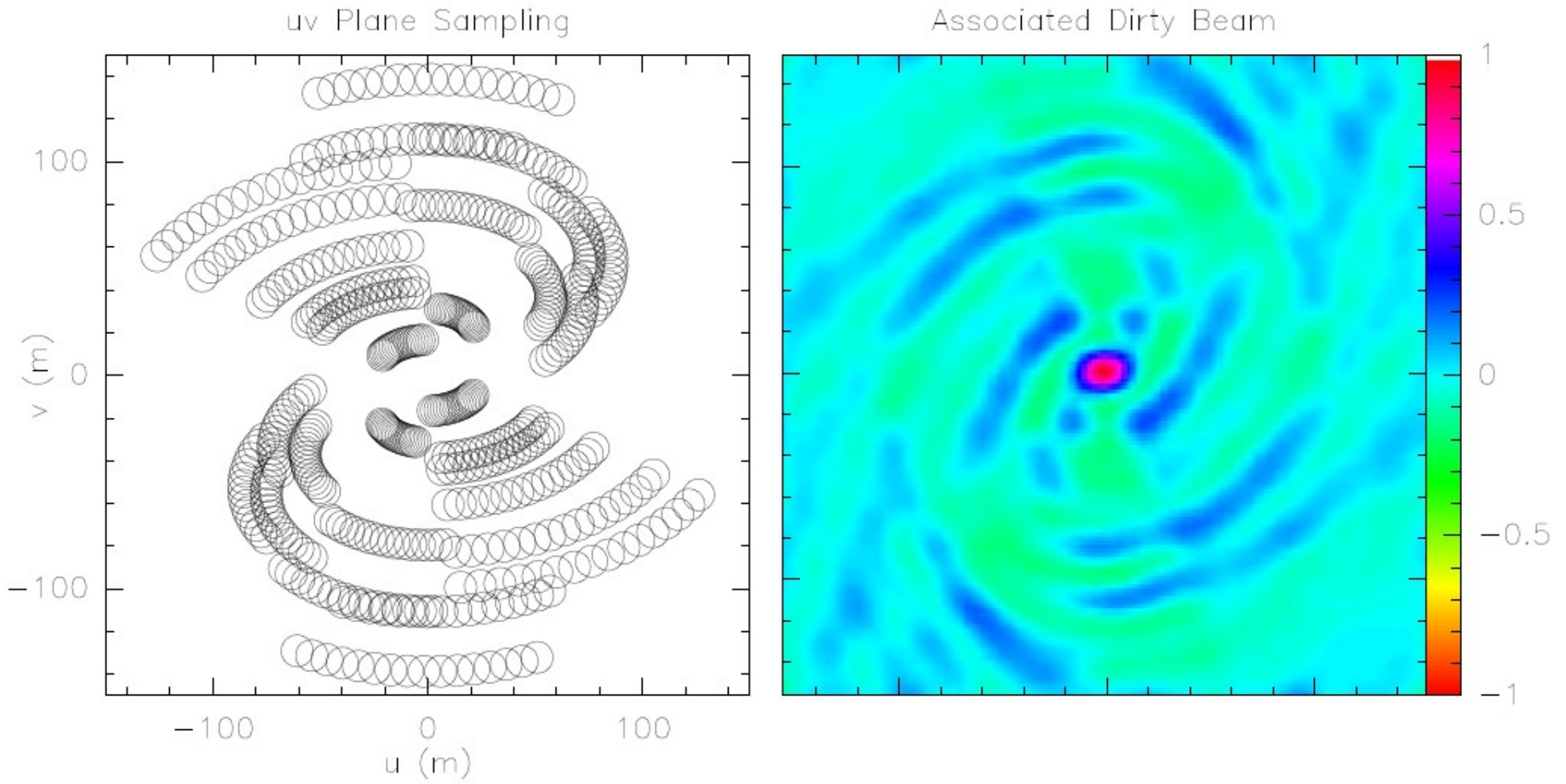
Dirty Beam Shape and Super Synthesis



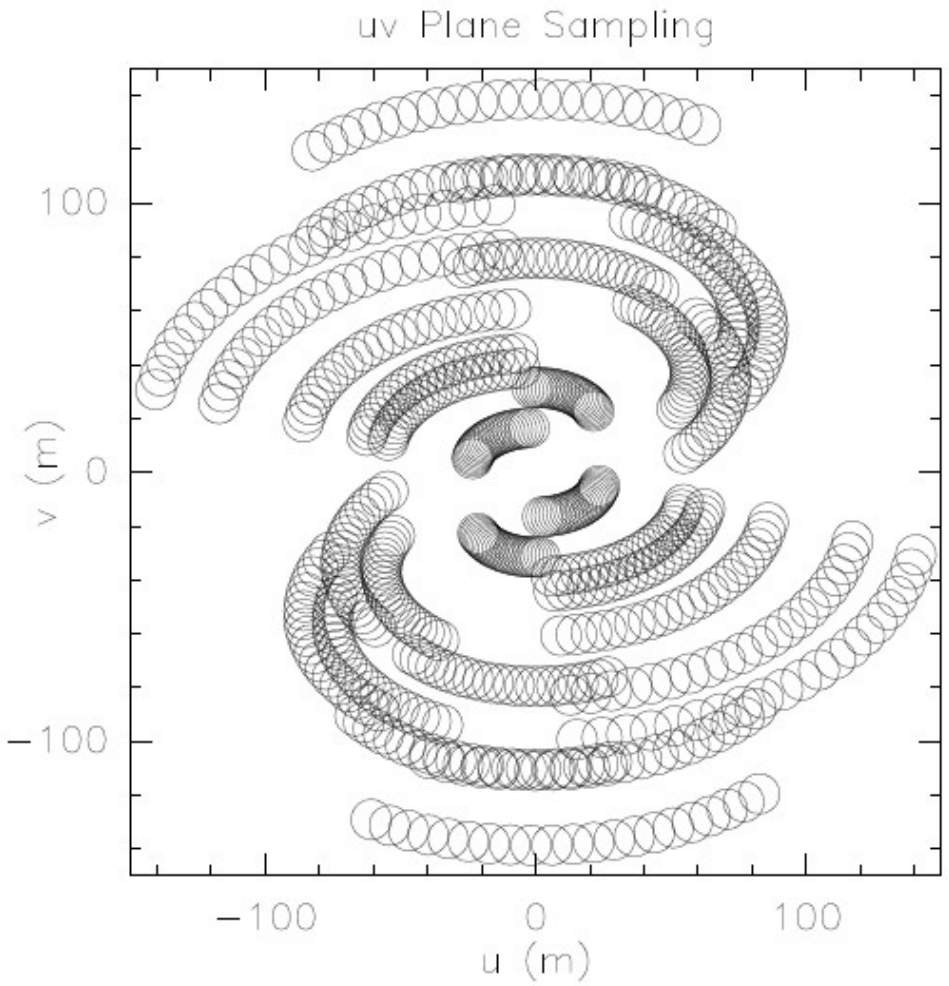
Dirty Beam Shape and Super Synthesis

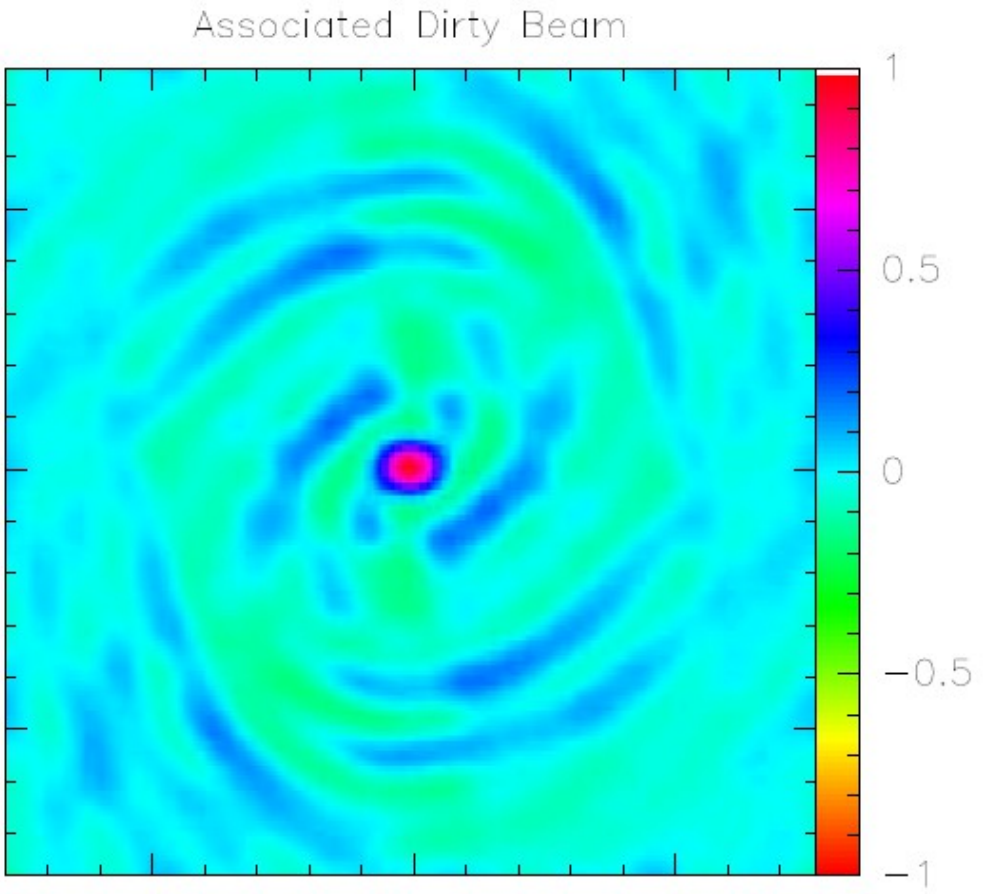


Dirty Beam Shape and Super Synthesis

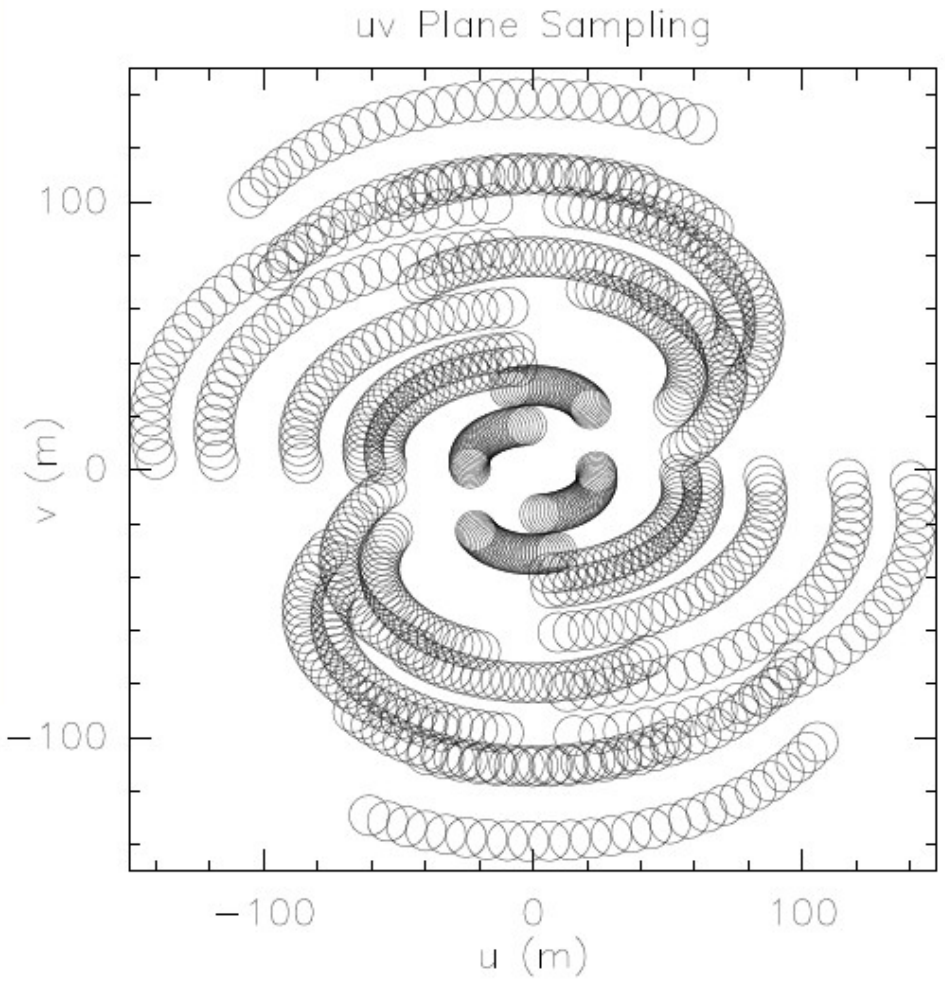


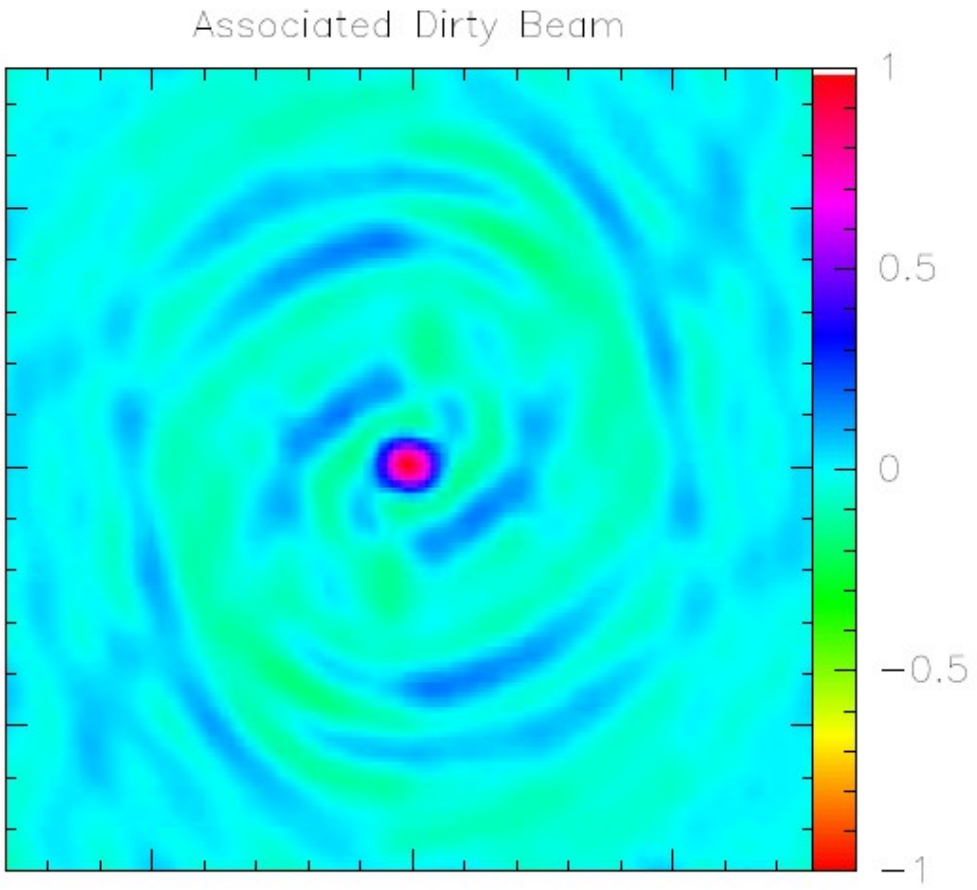
Dirty Beam Shape and Super Synthesis



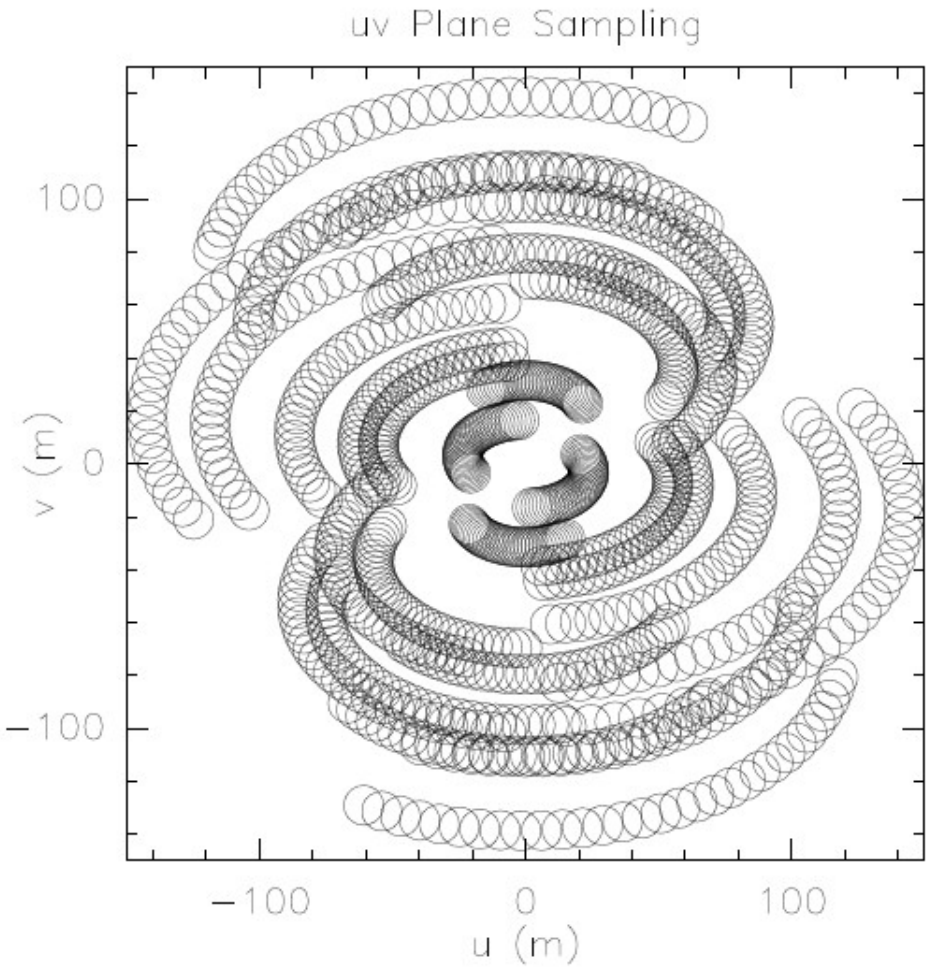


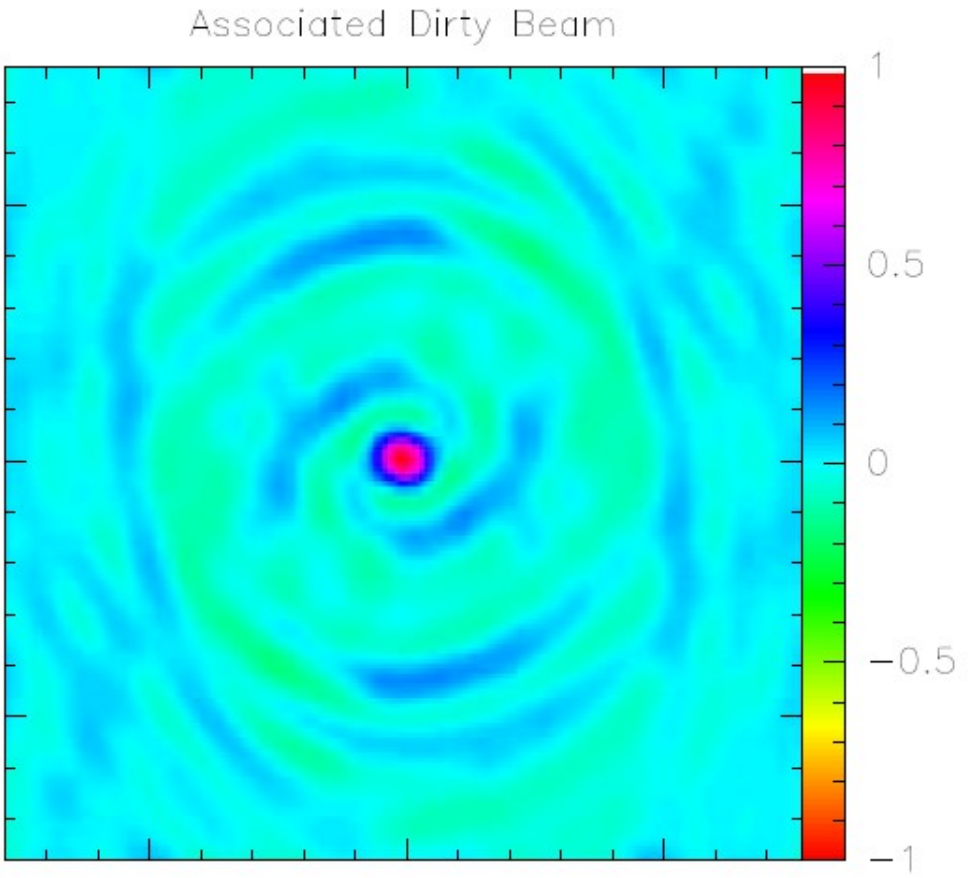
Dirty Beam Shape and Super Synthesis



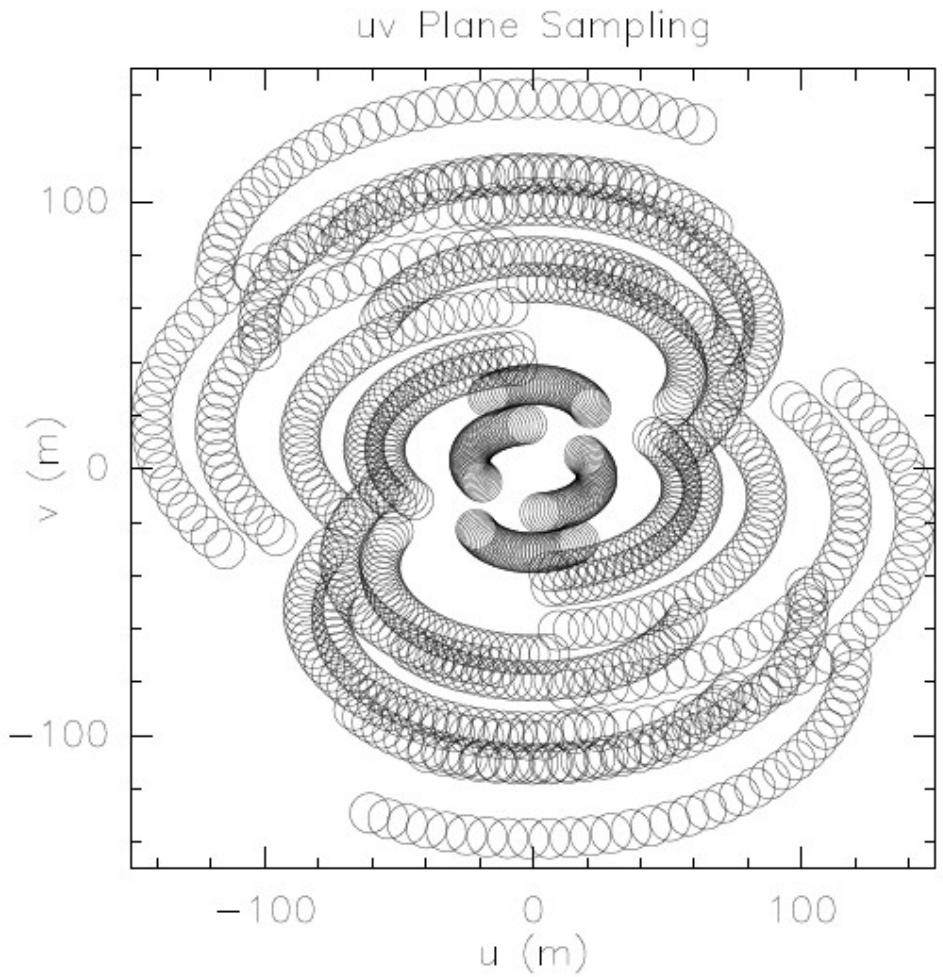


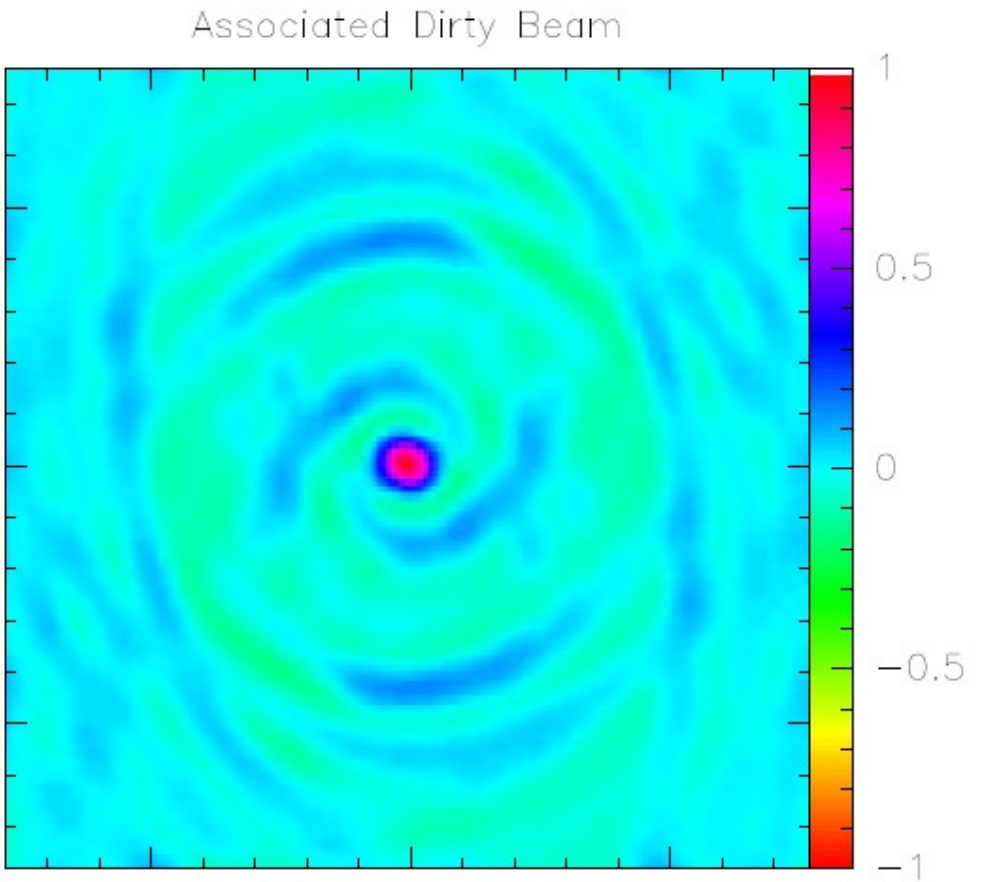
Dirty Beam Shape and Super Synthesis





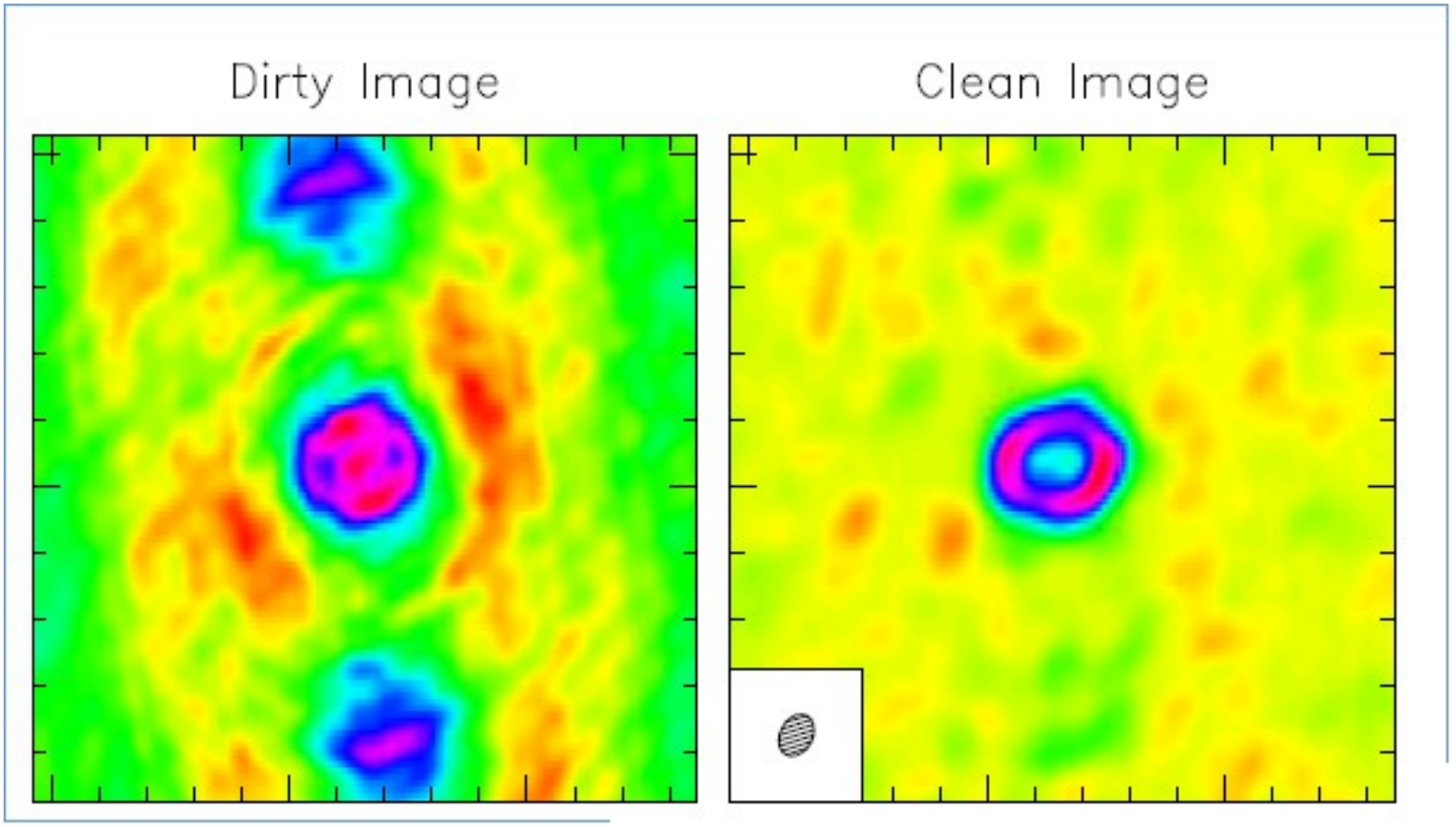
Dirty Beam Shape and Super Synthesis





The sampling is never perfect.

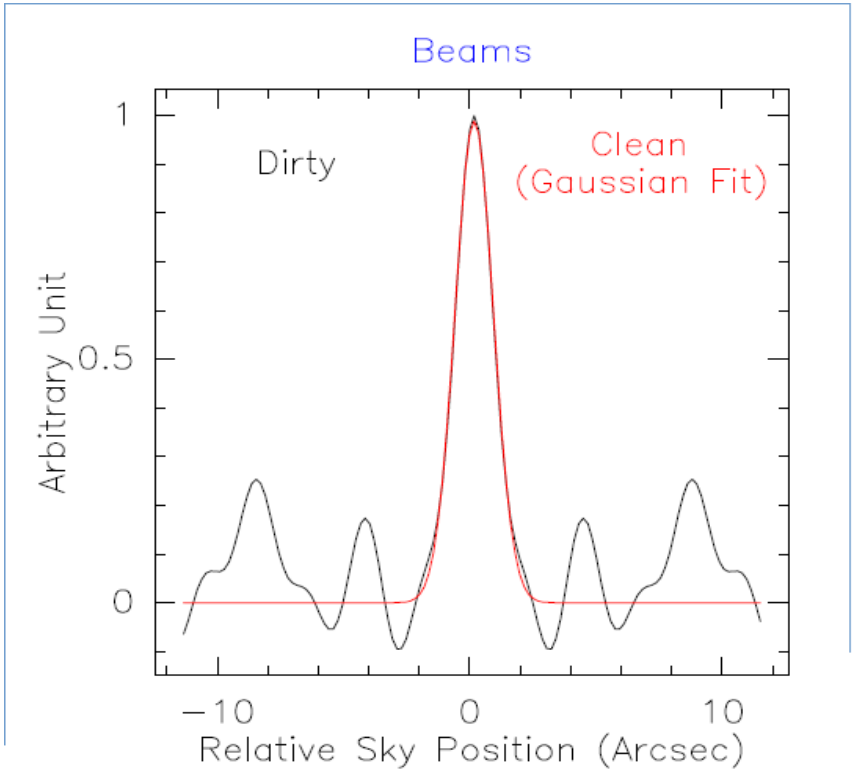
Deconvolution to make a clean image compatible with the sky intensity distribution



Deconvolution – The Basic CLEAN Algorithm

A priori assumption:

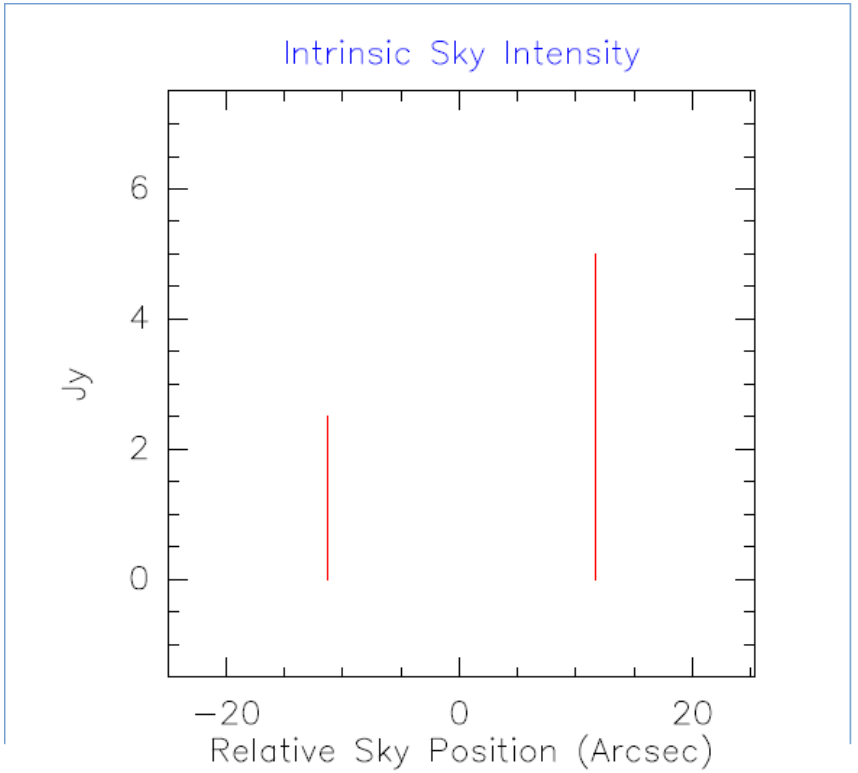
source = collection of point sources



Deconvolution – The Basic CLEAN Algorithm

A priori assumption:

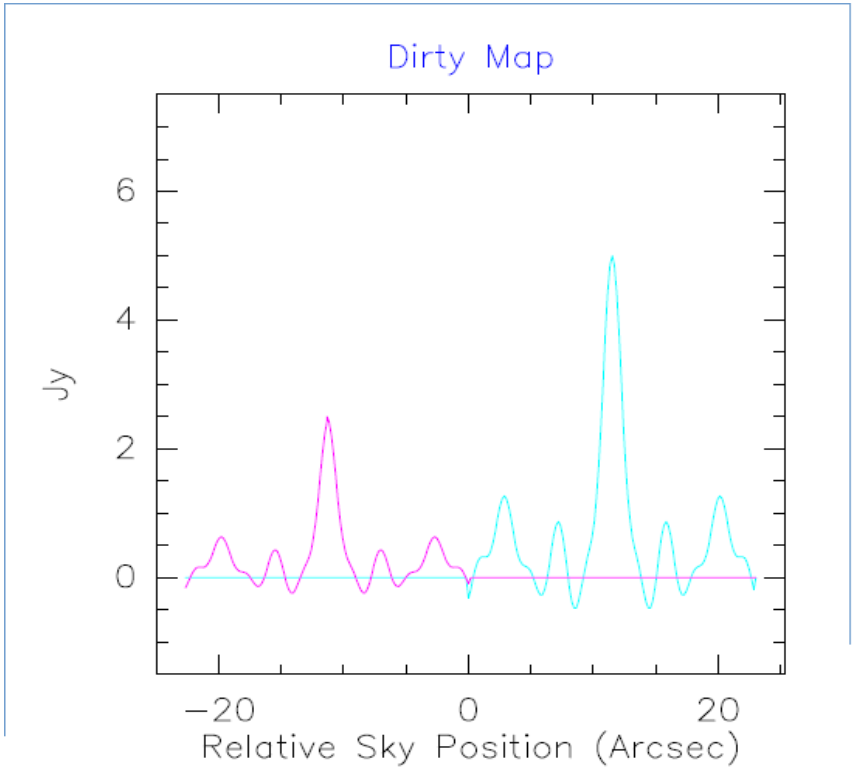
source = collection of point sources



Deconvolution – The Basic CLEAN Algorithm

A priori assumption:

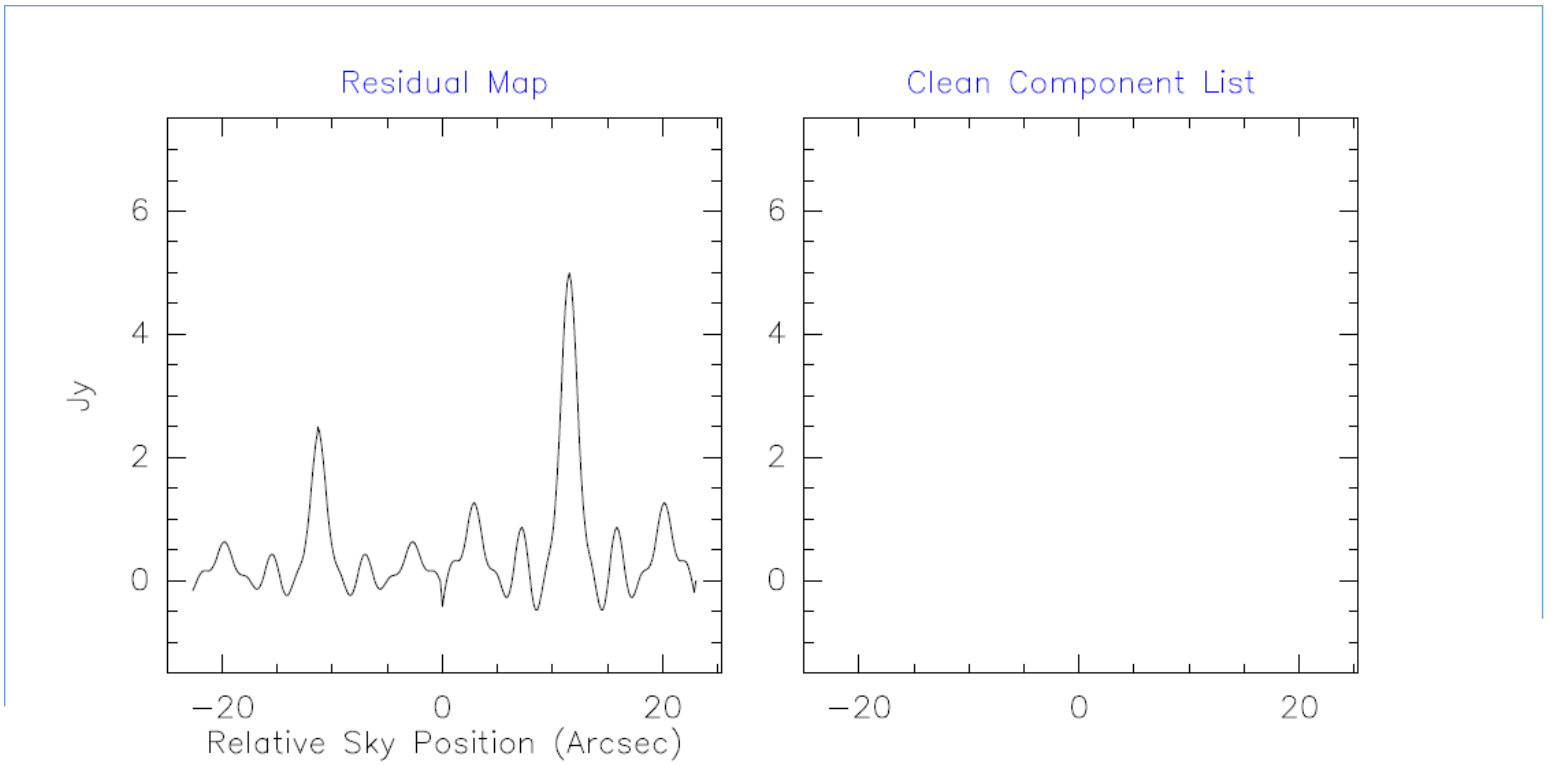
source = collection of point sources



Deconvolution – The Basic CLEAN Algorithm

Algorithm:

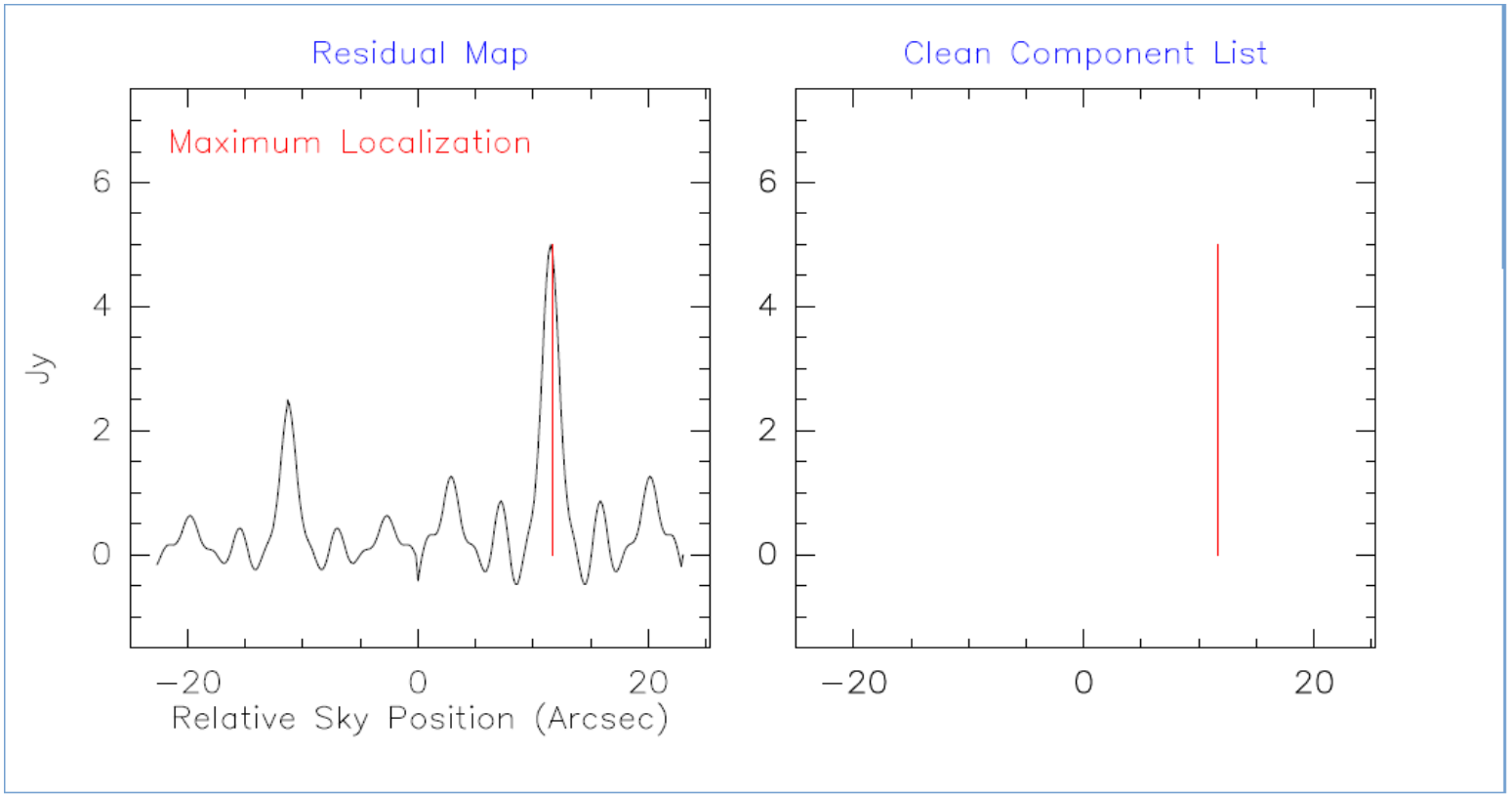
1. Initialize the residual map to the dirty map; the Clean component list to an empty value



Deconvolution – The Basic CLEAN Algorithm

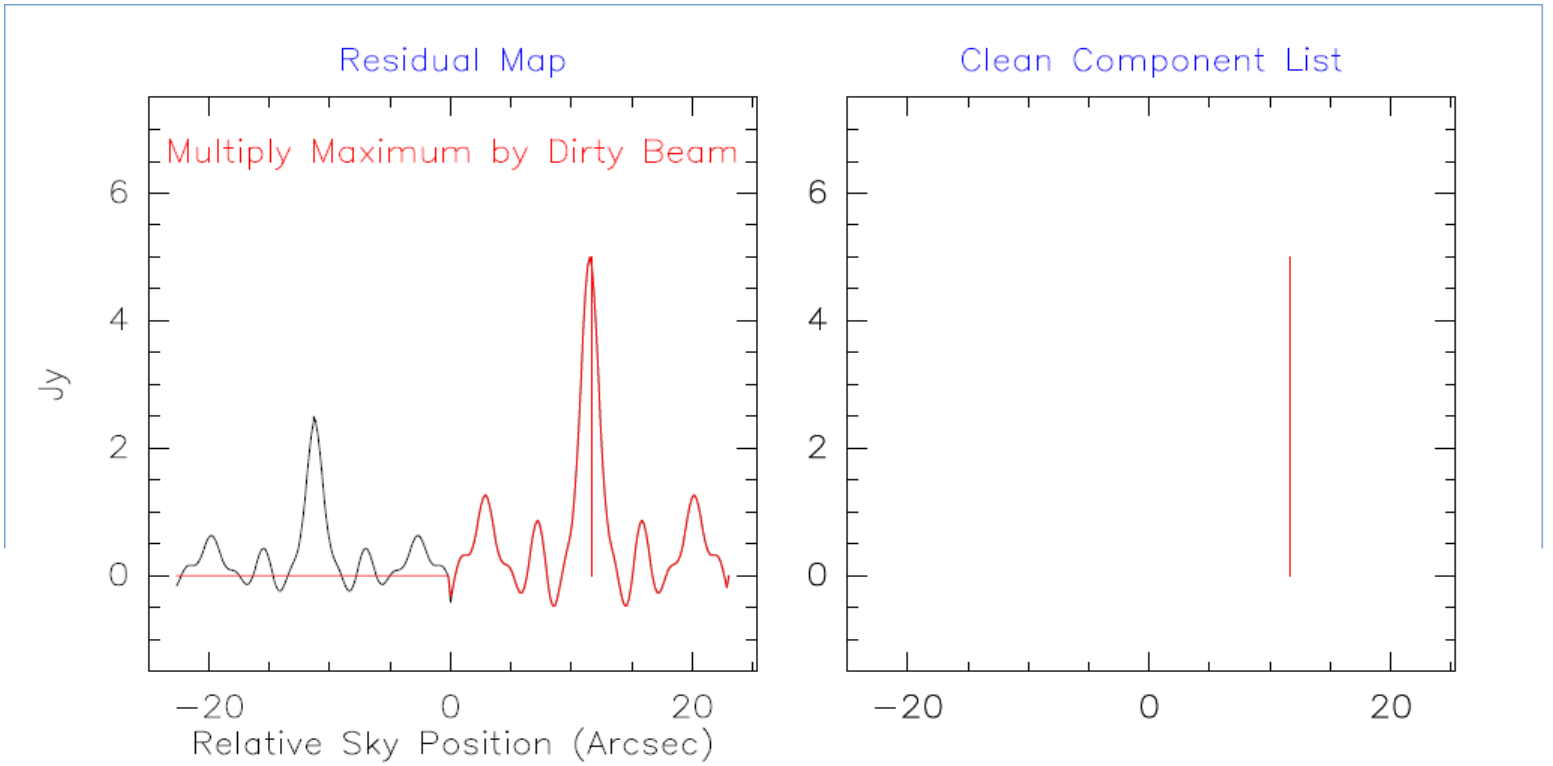
Algorithm:

2. Identify pixel of I_{max} in residual map as a point source
3. Add $\gamma \cdot I_{max}$ to clean component list (here $\gamma=1$)



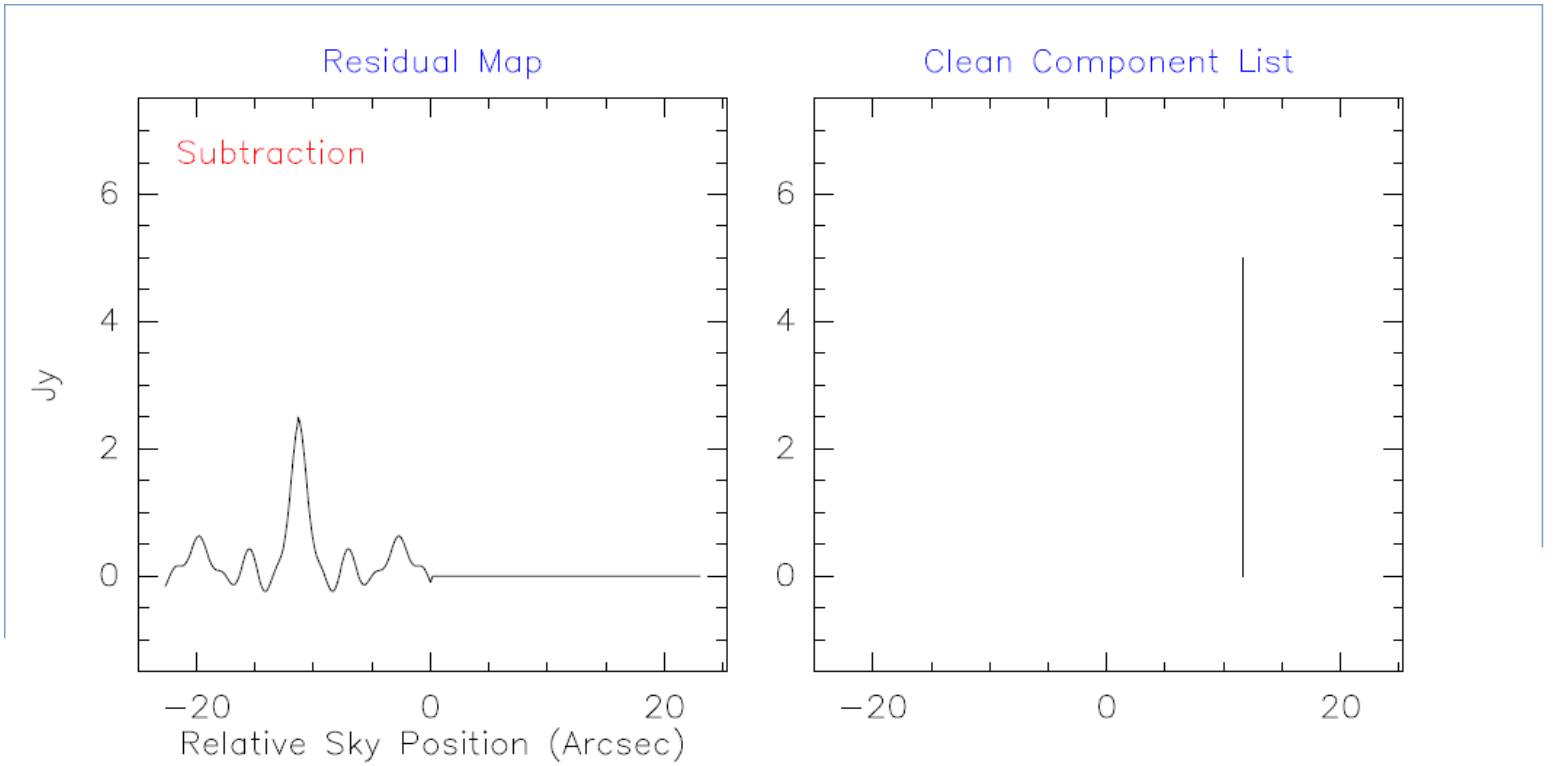
Deconvolution – The Basic CLEAN Algorithm

Algorithm:
4. multiply maximum by dirty beam



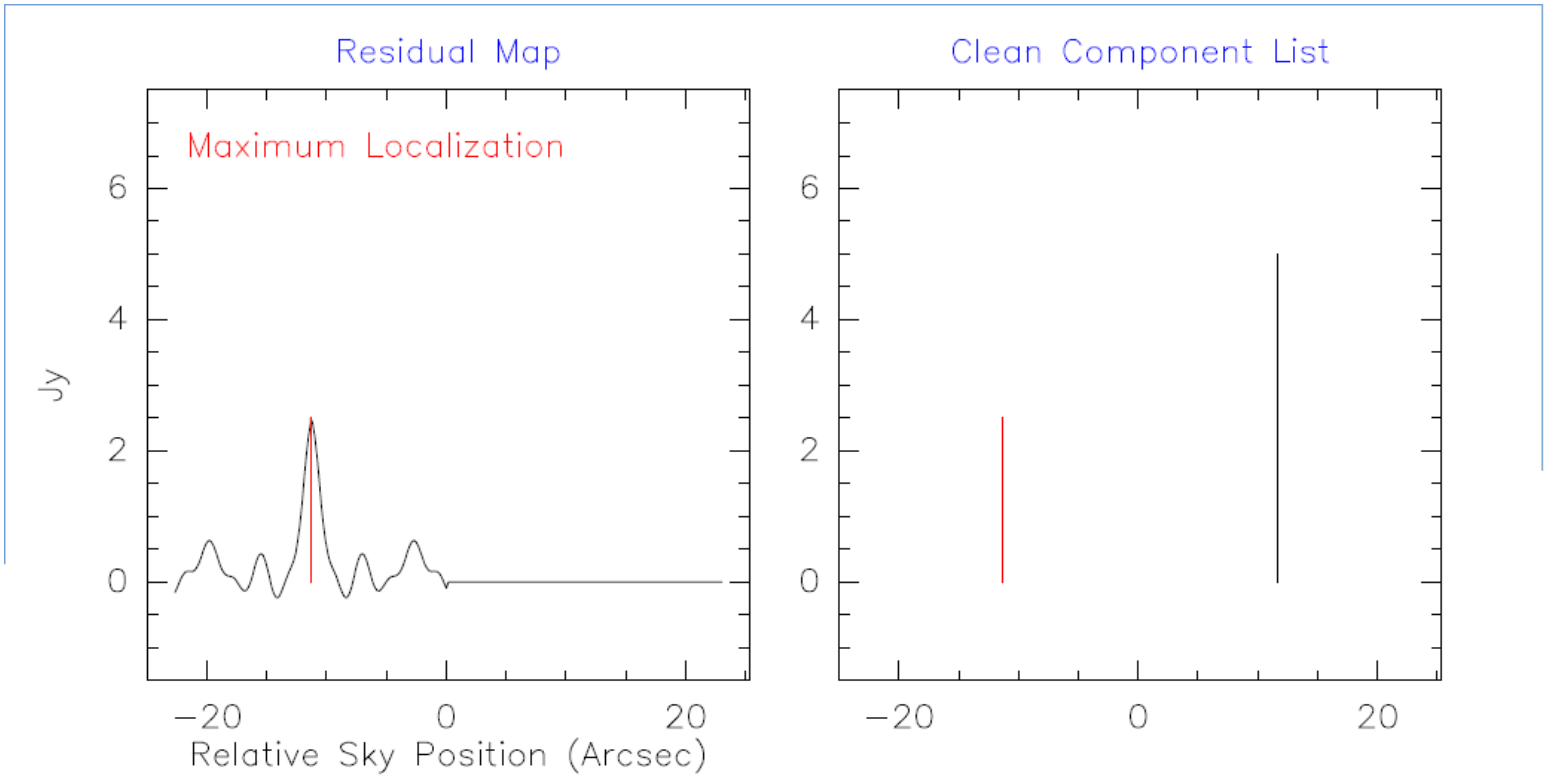
Deconvolution – The Basic CLEAN Algorithm

Algorithm:
5. subtract from residual



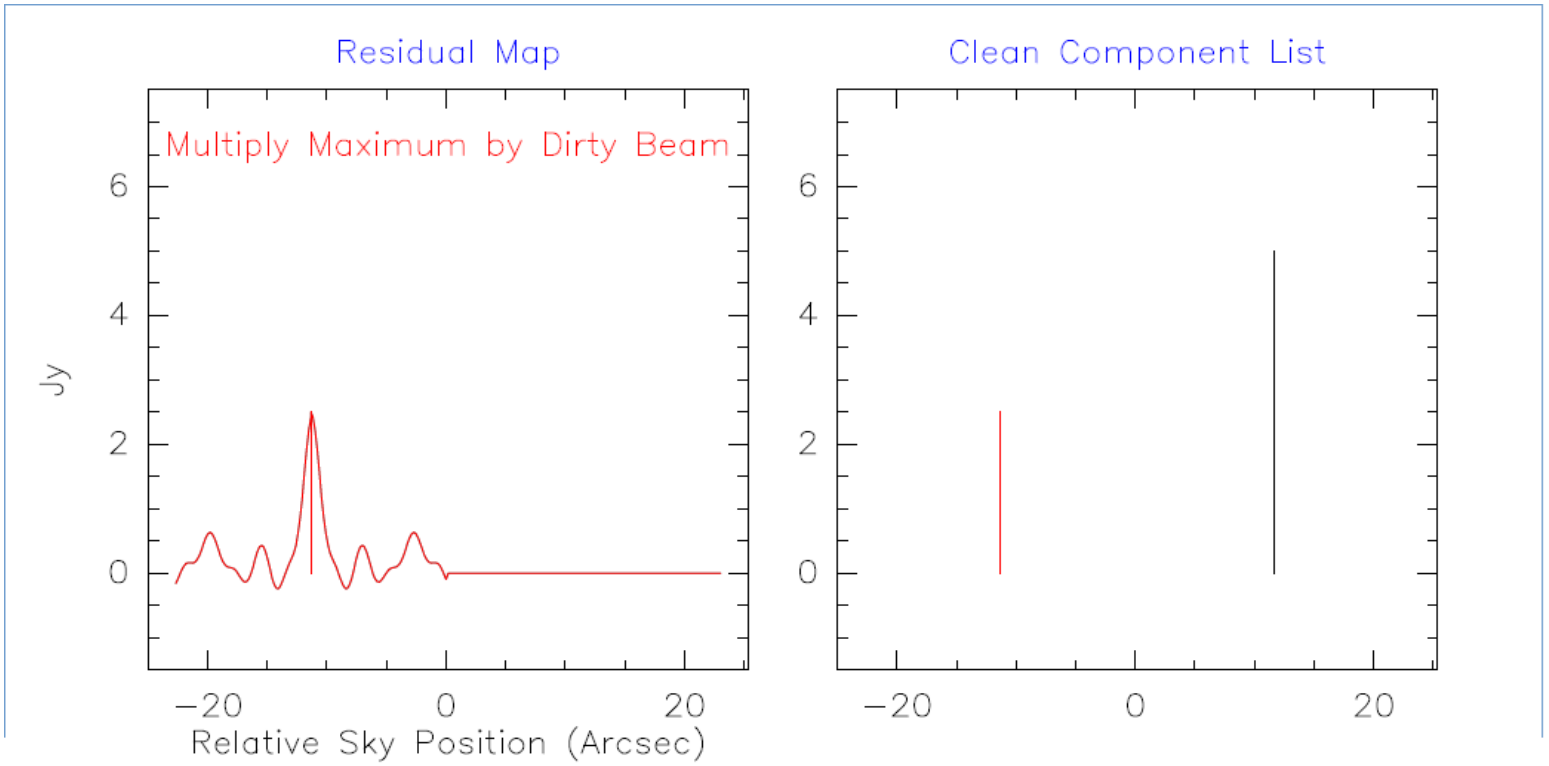
Deconvolution – The Basic CLEAN Algorithm

Algorithm:
6. Iterate from point 2



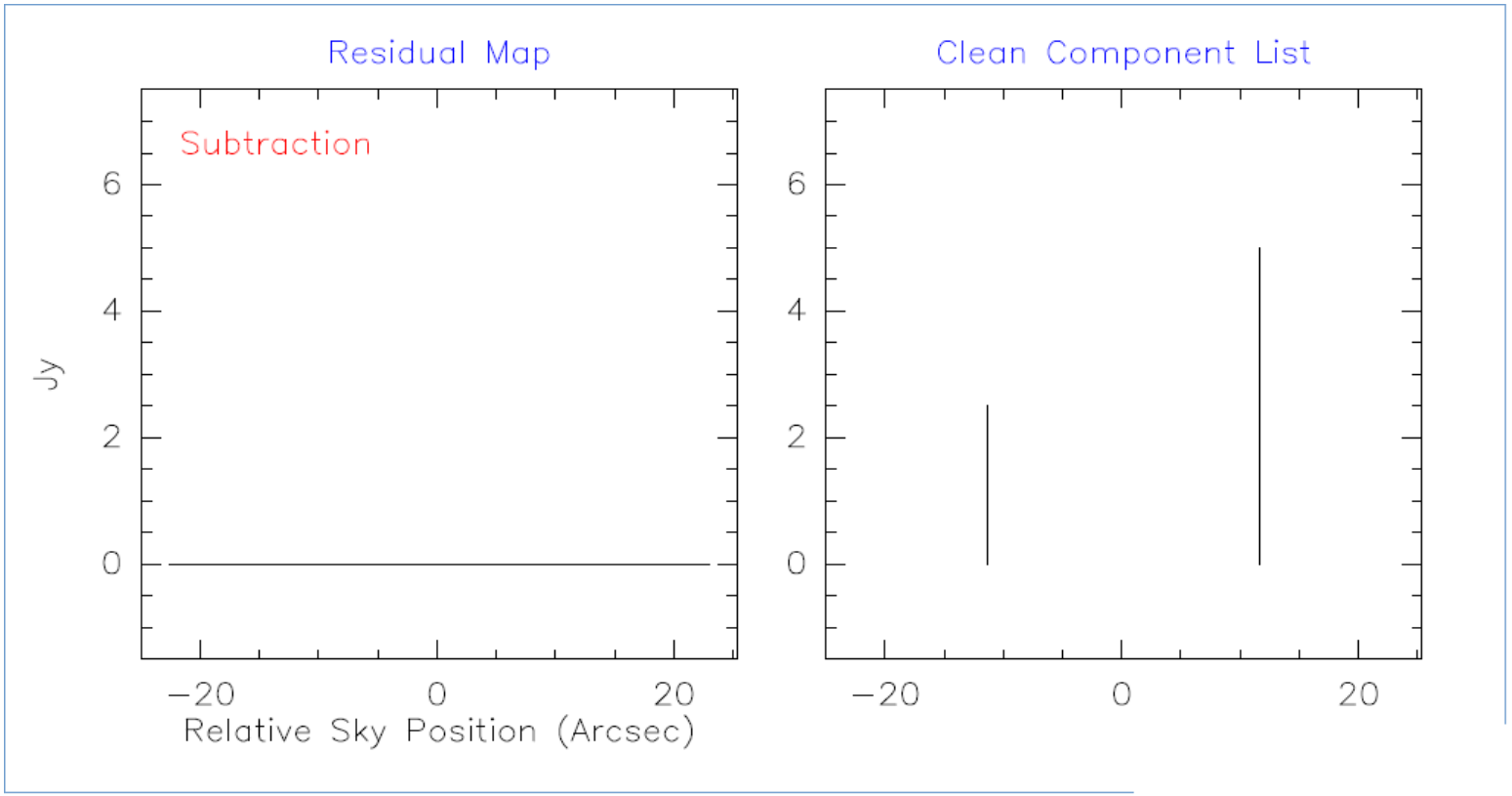
Deconvolution – The Basic CLEAN Algorithm

Algorithm:
6. Iterate from point 2



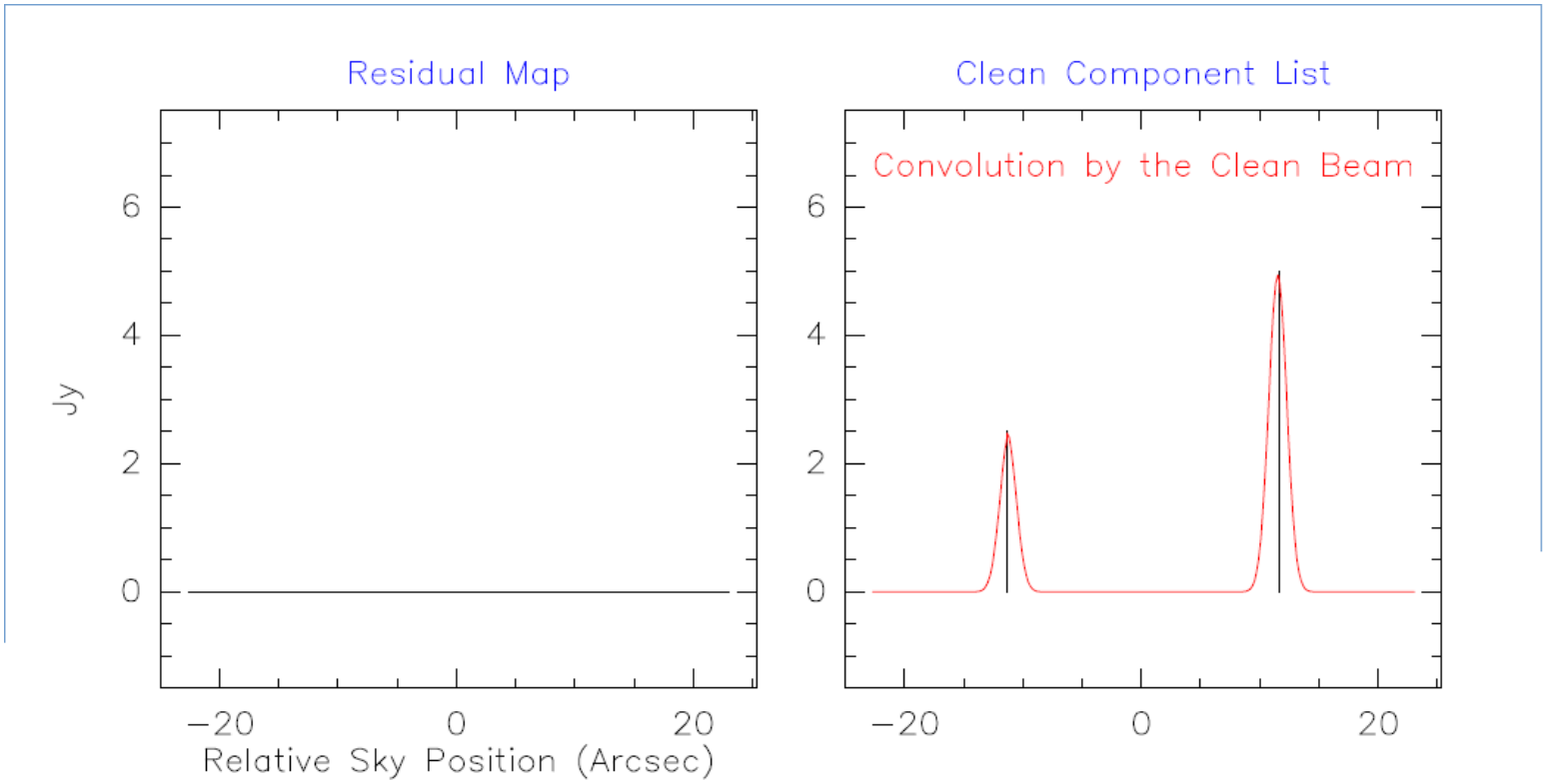
Deconvolution – The Basic CLEAN Algorithm

Algorithm:
6. Iterate from point 2



Deconvolution – The Basic CLEAN Algorithm

Algorithm:
7. Convolution by Clean beam



Deconvolution – The Basic CLEAN Algorithm

Stopping criterions:

- **Number of clean components**
- **$|I_{max}|$ < of noise**
- **$|I_{max}|$ < fraction of dirty map max**

Loop gain: good results when γ ~ 0.1 – 0.3

To speed the algorithm is possible to give a priori information about where to search for clean components **(“clean boxes”)**
useful but potentially dangerous!

Deconvolution – CLEAN Variants

Basic:
Hogbom (Hogbom 1974)

Faster Search algorithms:
Clark (Clark 1980)
MX (Cotton & Schwab 1984)

Better handling of extended sources:
Multi Scale Clean (Cornwell 1998)

CLEAN in CASA

Maybe the task with the longest list of inputs
For the moment we will use just few of them!

Mode= **mfs** (multi frequency synthesis) produces one image for all the specified data combined
= channel produces a “cube” image with different planes corresponding to the channels specified by **start, nchan, width**

Start = first channel to include
Nchan = number of channels
Width = output channel width
default=1; >1 indicates channel averaging

CLEAN in CASA

Imsize=
cellsize =

Image size and pixel size chosen in order to accurately sample the resolution element and cover the primary beam

Resolution element: $\theta \sim \lambda/B_{max}$

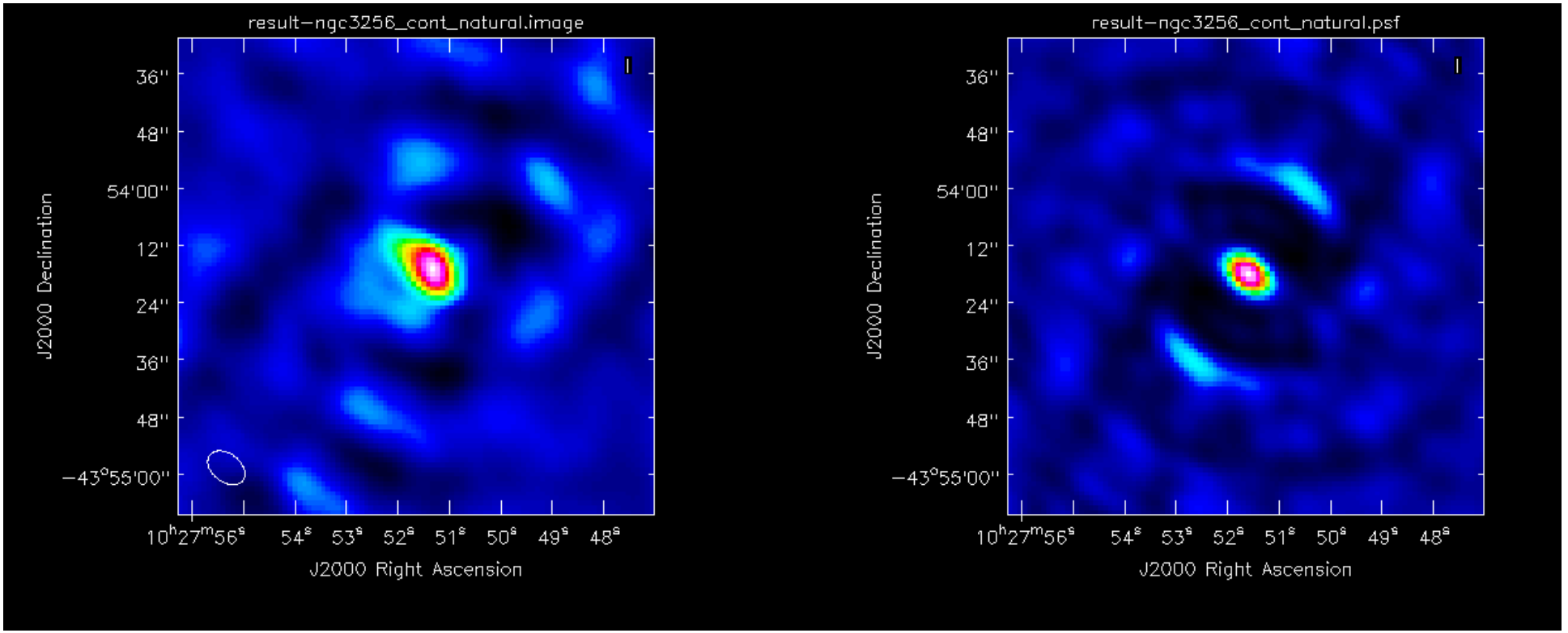
"Accurately sample" means having at least 3-4 pixel in each resolution element

Primary beam: $\sim\lambda/D$

cellsize = '1arcsec' arcsec per pixel
imsize = 100 in pixel

CLEAN in CASA

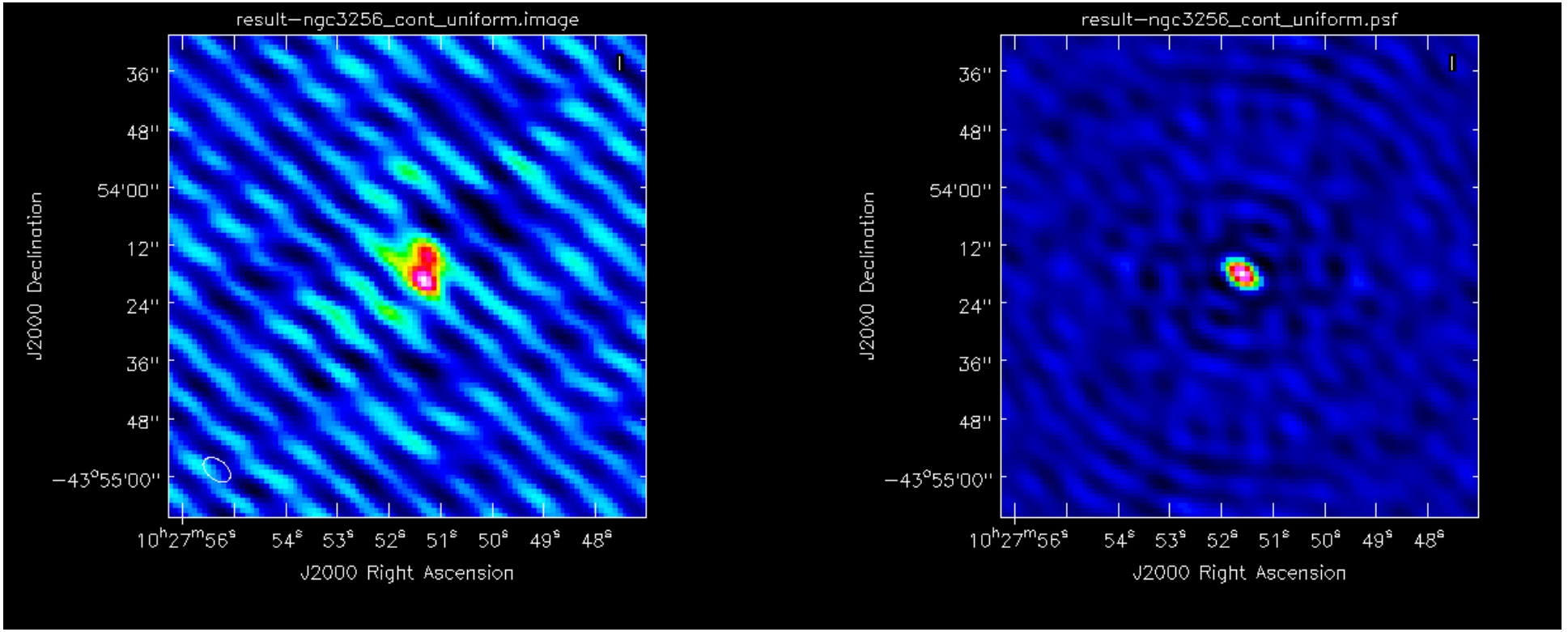
weighting = **natural** (default)



Synthesized beam= 8.76 x 5.89 arcsec

CLEAN in CASA

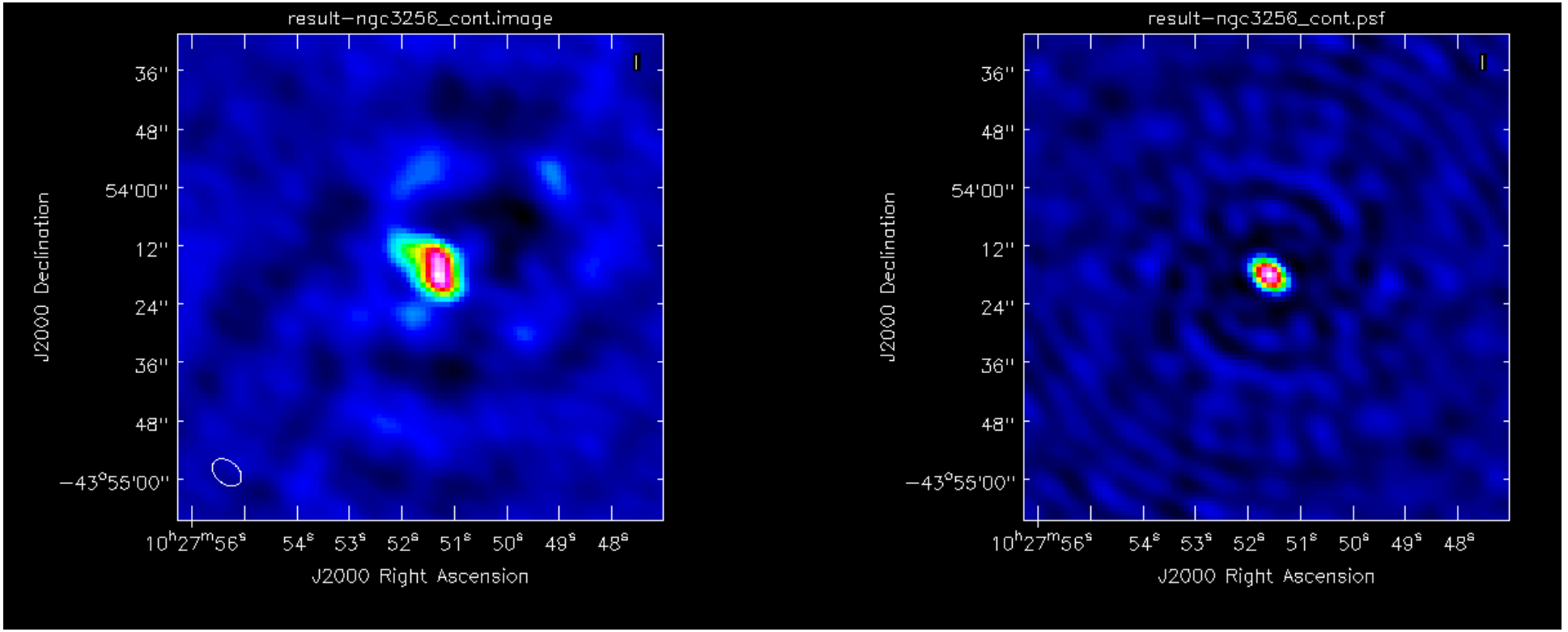
weighting = uniform



Synthesized beam= 6.51 x 4.02 arcsec

CLEAN in CASA

weighting = **briggs ; robust =0** something in between



Synthesized beam= 6.77 x 4.61 arcsec

CLEAN in CASA

niter = **500**

maximum number of clean iteration

threshold = '0.5mJy'

stop cleaning if the maximum residual is below this value (1.5-2 times rms expected)

mask=[]

it is possible to give the region where to search for components

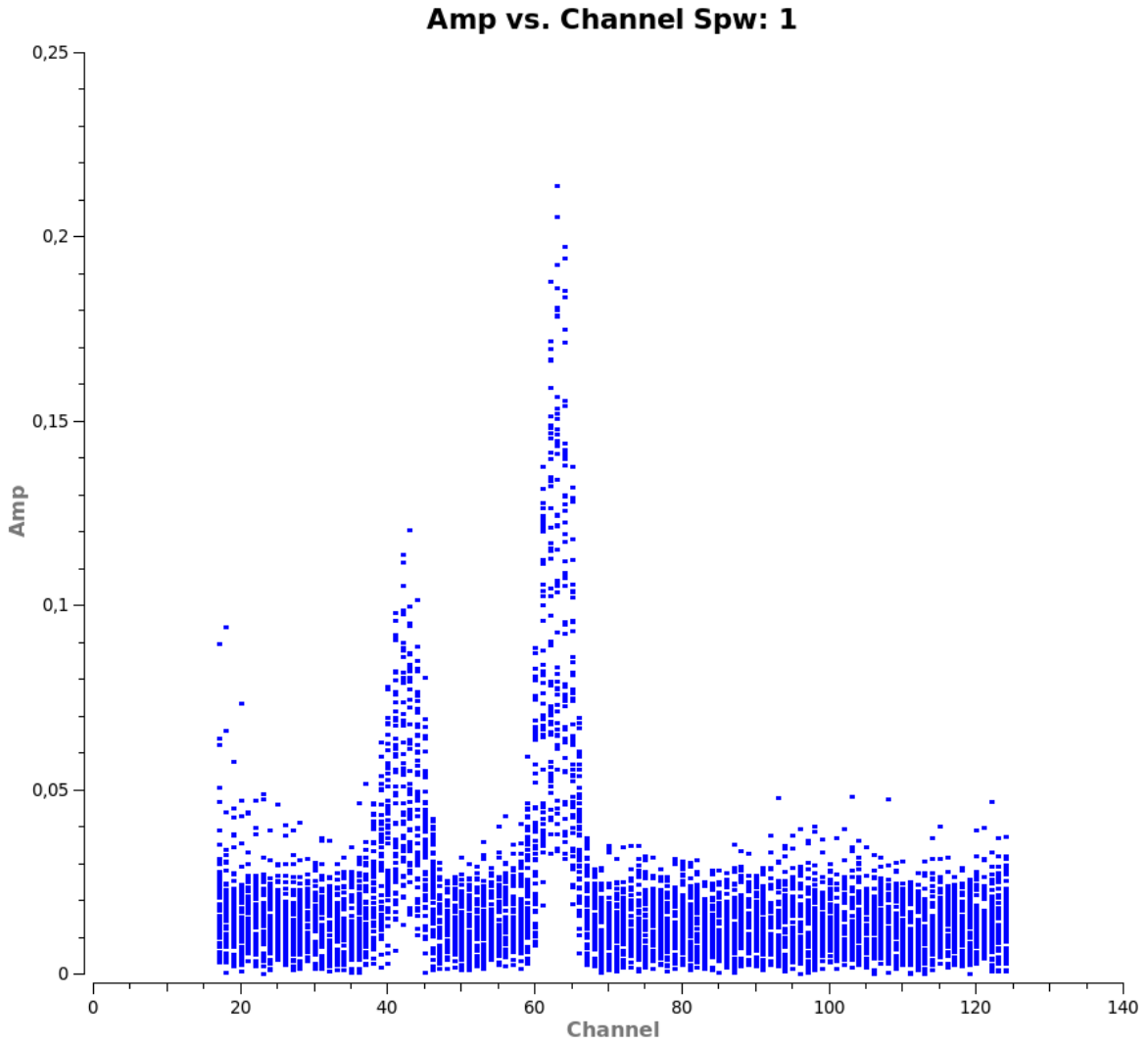
or do it interactively:

interactive=T

NGC3256 continuum image

Dataset: **ngc3256_line_target.ms**

plot amp vs frequency (channel)
and identify
line-free channels
in each spectral window



NGC3256 continuum image

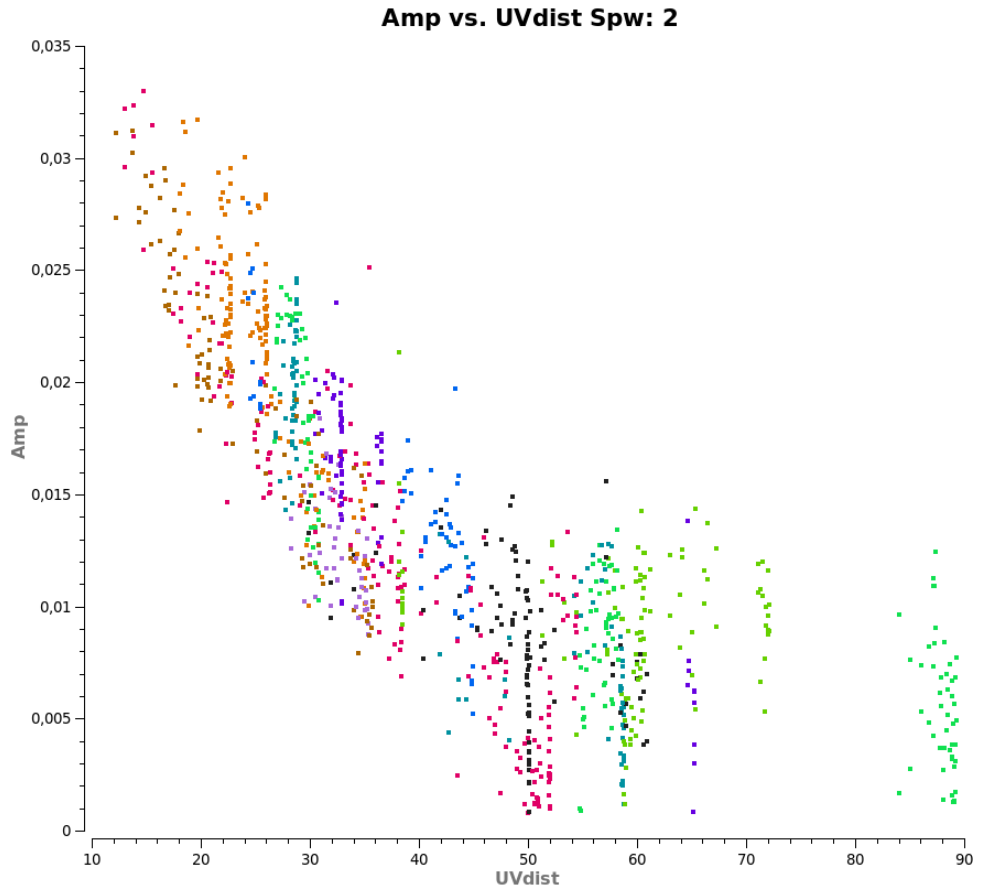
Dataset**: ngc3256_line_target.ms**

Spw 0 : 20~53; 71~120
Spw 1 : 70~120
Spw 2 : 20~120
Spw 3 : 20~120

NGC3256 continuum image

Dataset: **ngc3256_line_target.ms**

Amp vs uvdist in spw 2



Continuum emission is....

NGC3256 continuum image

Dataset: **ngc3256_line_target.ms**

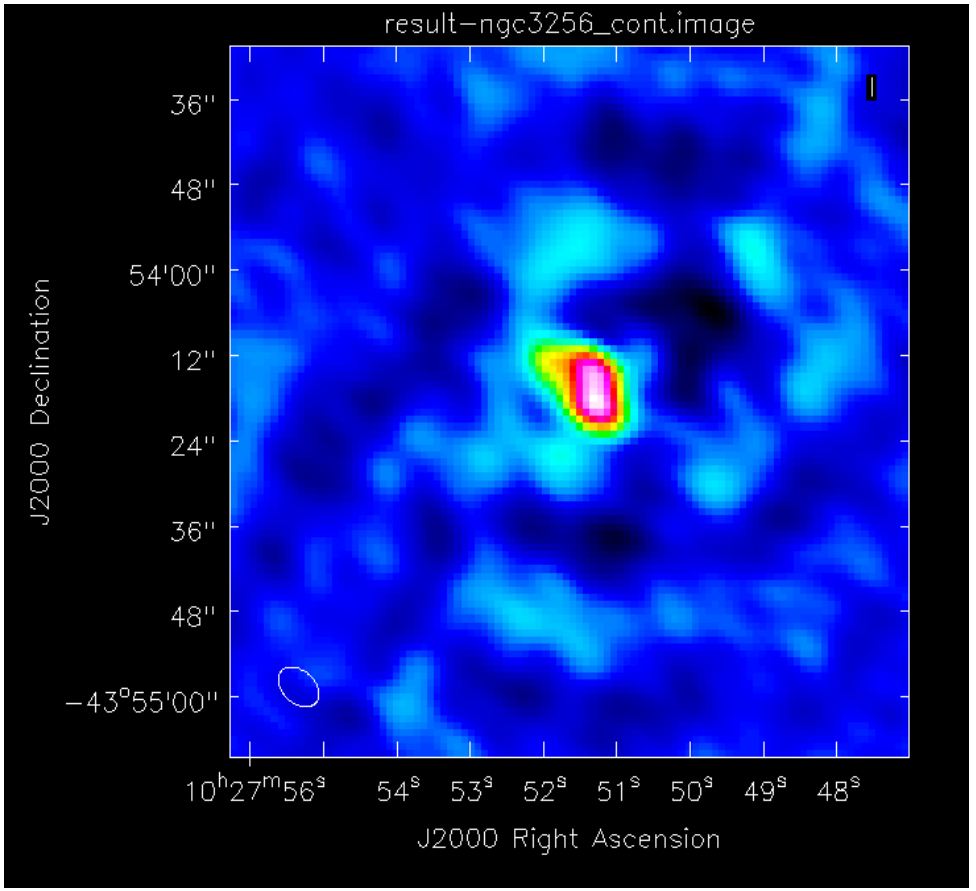
clean input:

```
os.system('rm -rf result-ngc3256_cont*')
clean( vis='ngc3256_line_target.ms', imagename='result-
ngc3256_cont',
spw='0:20~53;71~120,1:70~120,2:20~120,3:20~120',
psfmode='hogbom', mode='mfs', niter=500,
threshold='0.3mJy', mask=[42,43,59,60],
imsize=100, cell='1arcsec', weighting='briggs',
robust=0.0, interactive=False, usescratch=False)
```

NGC3256 continuum image

Dataset**: ngc3256_line_target.ms**

result-ngc3256-cont.image:



NGC3256 subtract continuum emission

uvcontsub(vis = 'ngc3256_line_target.ms',
fitspw='0:20~53;71~120,1:70~120,2:20~120,3:20~120', solint ='int',
fitorder = 1,combine='spw')

fitspw = to specify the channels to be used in the fit for the continuum

solint = timescale for the per-baseline fit

fitorder = order of the polynomial to fit the continuum

combine = spw to form a spw-merged continuum estimate

It produces a new dataset **.contsub**

NGC3256 CO line emission

Dataset: **ngc3256_line_target.ms.uvcontsub**

clean input:

```
os.system('rm -rf result-ngc3256_line_CO.*')
clean(vis='ngc3256_line_target.ms.contsub',
imagename='result-ngc3256_line_CO',
spw='0:38~87', mode='channel', start='', nchan=50, width='',
psfmode='hogbom', outframe='BARY',
restfreq='115.271201800GHz',
mask=[53,50,87,83], niter=500, interactive=T, imsize=128,
cell='1arcsec', weighting='briggs', robust=0.0,
threshold='5mJy', usescratch=False)
```

NGC3256 CN(1-0),J=3/2-1/2 line emission

Dataset: **ngc3256_line_target.ms.uvcontsub**

clean input:

```
os.system('rm -rf result-ngc3256_line_CNhi.*')
clean(vis='ngc3256_line_target.ms', imagename='result-
ngc3256_line_CNhi',
outframe='LSRK', spw='1:50~76', start='', nchan=27,
width='', restfreq='113.48812GHz', selectdata=T,
mode='channel', niter=500, gain=0.1, psfmode='hogbom',
mask=[53,50,87,83], interactive=True, imsize=128,
cell='1arcsec', weighting='briggs', robust=0.0,
threshold='2mJy', usescratch=False)
```

NGC3256 CN(1-0),J=1/2-1/2 line emission

Dataset: **ngc3256_line_target.ms.uvcontsub**

clean input:

```
os.system('rm -rf result-ngc3256_line_CNlo.*')
clean( vis='ngc3256_line_target.ms', imagename='result-
ngc3256_line_CNlo',
outframe='LSRK', spw='1:29~54', start='', nchan=26,
width='', restfreq='113.17049GHz', selectdata=T,
mode='channel', niter=300, gain=0.1, psfmode='hogbom',
mask=[53,50,87,83], interactive=True, imsize=128,
cell='1arcsec', weighting='briggs', robust=0.0,
threshold='2mJy', usescratch=False)
```