Imaging

NAASC data analysis workshop

Presentation borrowed heavily from David Wilner, Scott Schnee and Steve Myers

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

Outline

- theory of synthesis imaging : FTs and uv coverage
- demonstration of deconvolution a.k.a. clean
- casa clean:
  - basic image parameters
  - continuum & cubes
  - single fields & mosaicing
  - weighting
  - clean algorithms
  - stopping & interactive
  - multiscale
- image analysis & viewer
- uvcontsub

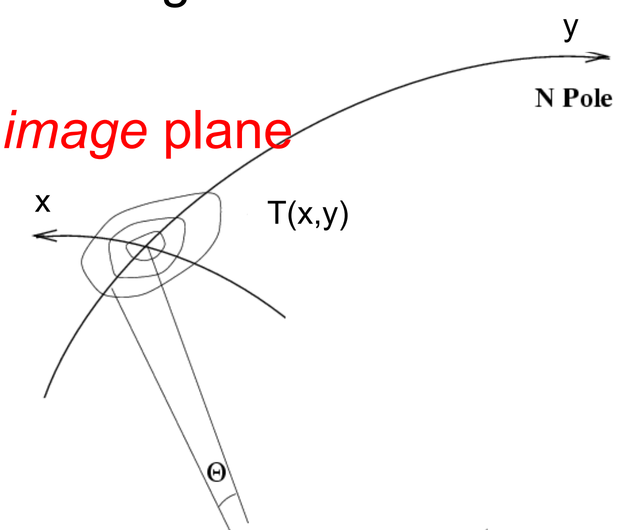
NRAO

From Sky Brightness to Visibility

1. An interferometer measures the interference pattern produced by two 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



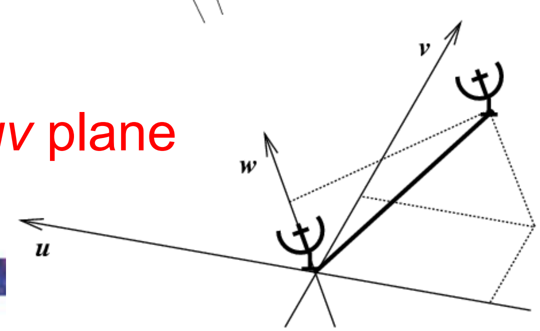
Fourier space/domain

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

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

Image space/domain

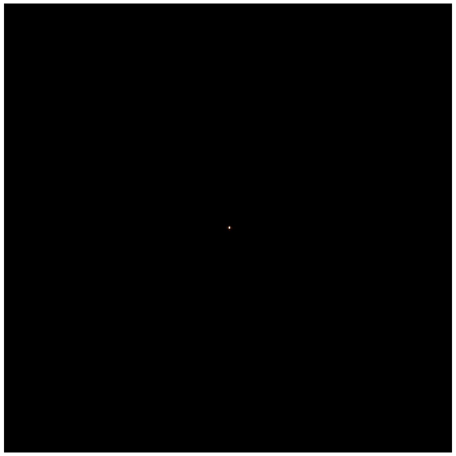
uv plane



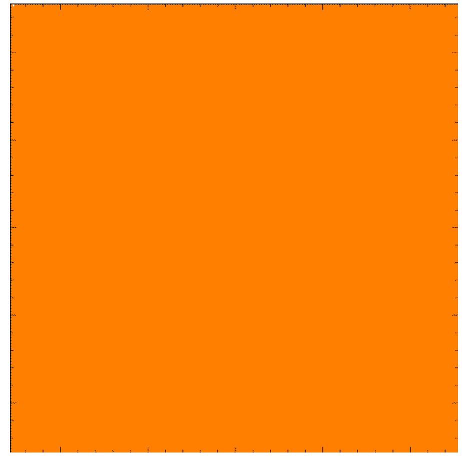
Some 2D Fourier Transform Pairs

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

δ Function

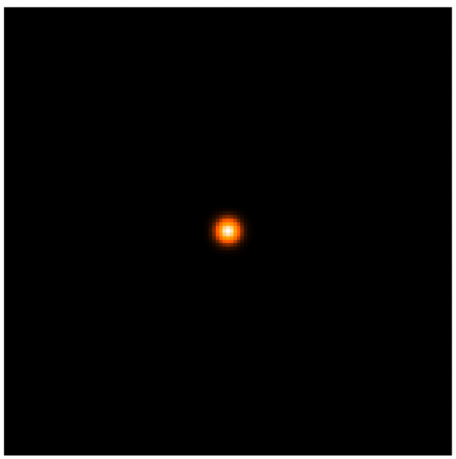


$\rightleftharpoons$

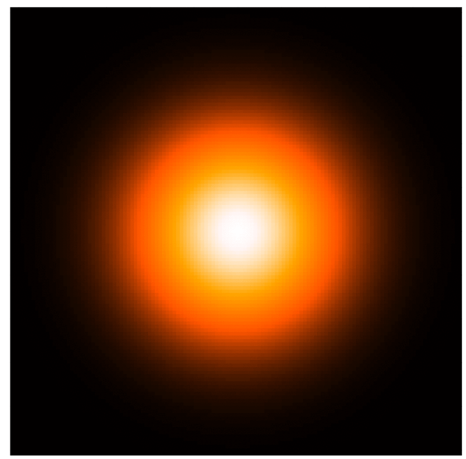


Constant

Gaussian



$\rightleftharpoons$



Gaussian

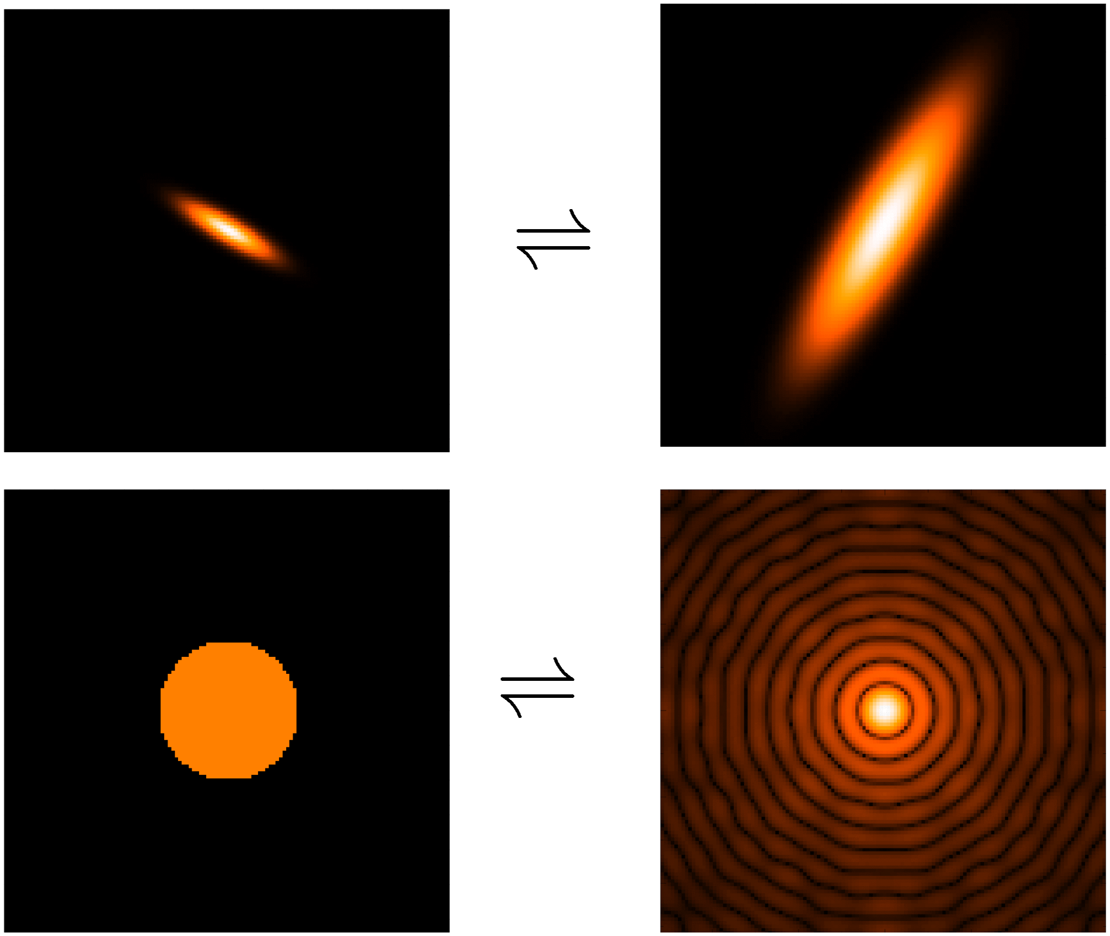
narrow features transform to wide features (and vice-versa)

More 2D Fourier Transform Pairs

T(x,y) — Amp{V(u,v)}

elliptical Gaussian ⇌ elliptical Gaussian

Disk ⇌ Bessel



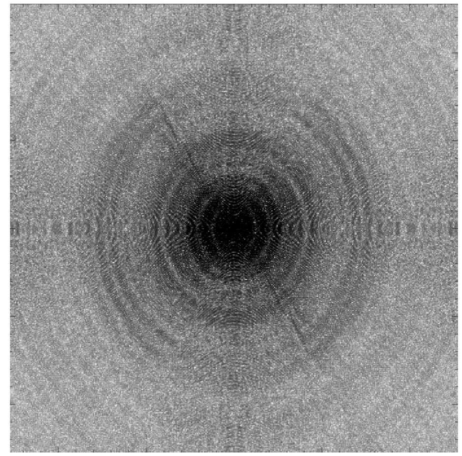
sharp edges result in many high spatial frequencies

More 2D Fourier Transform Pairs

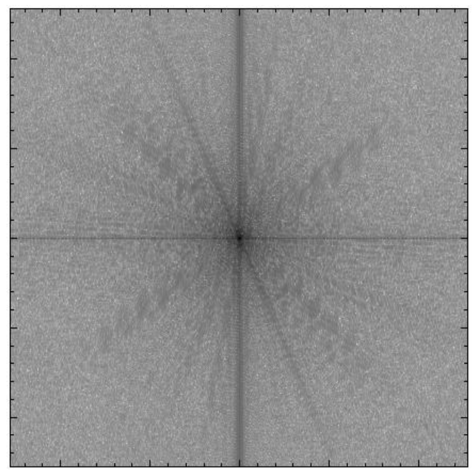
T(x,y)

Amp{V(u,v)}

$\rightleftharpoons$



$\rightleftharpoons$

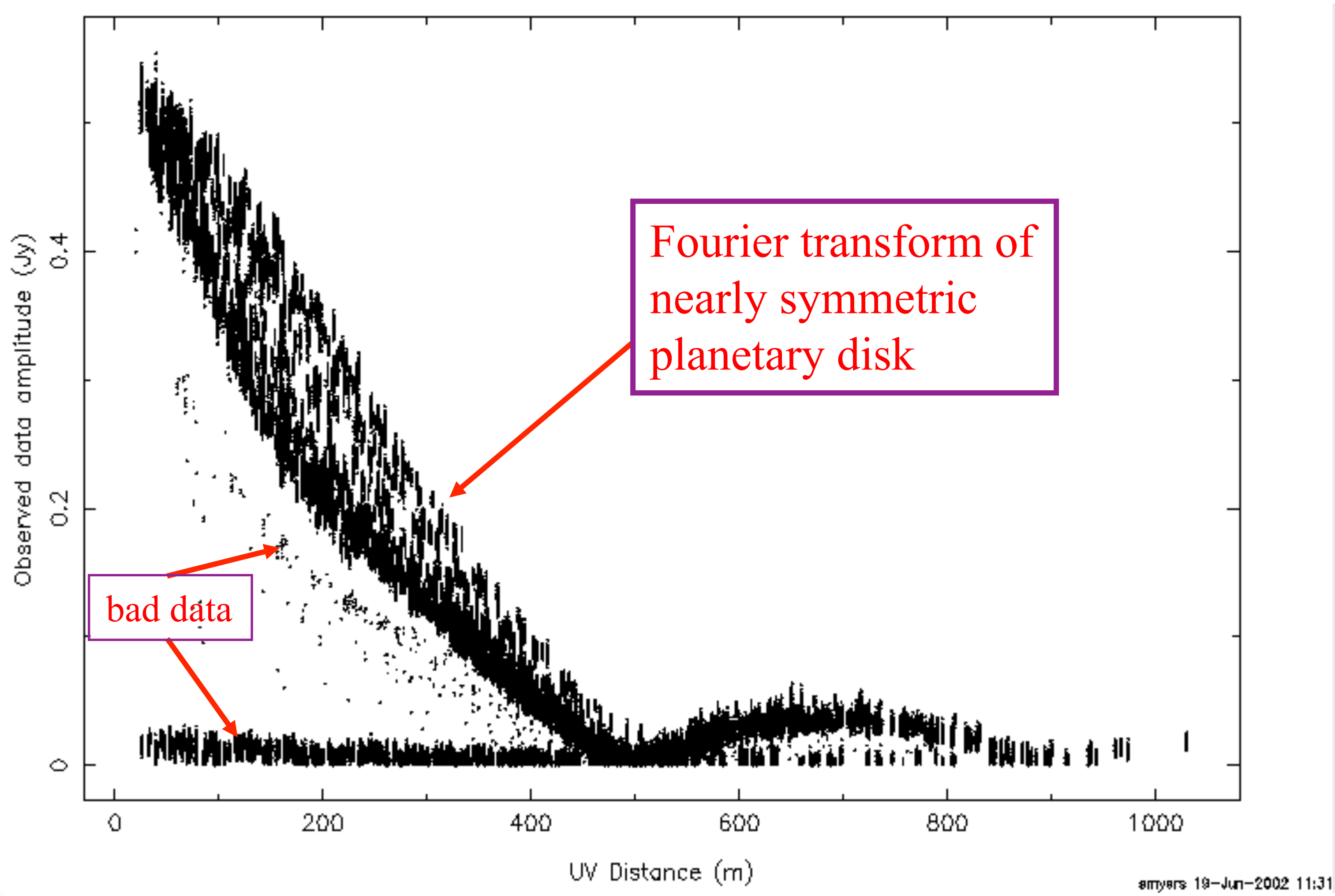


complicated structure on many scales

NRAO

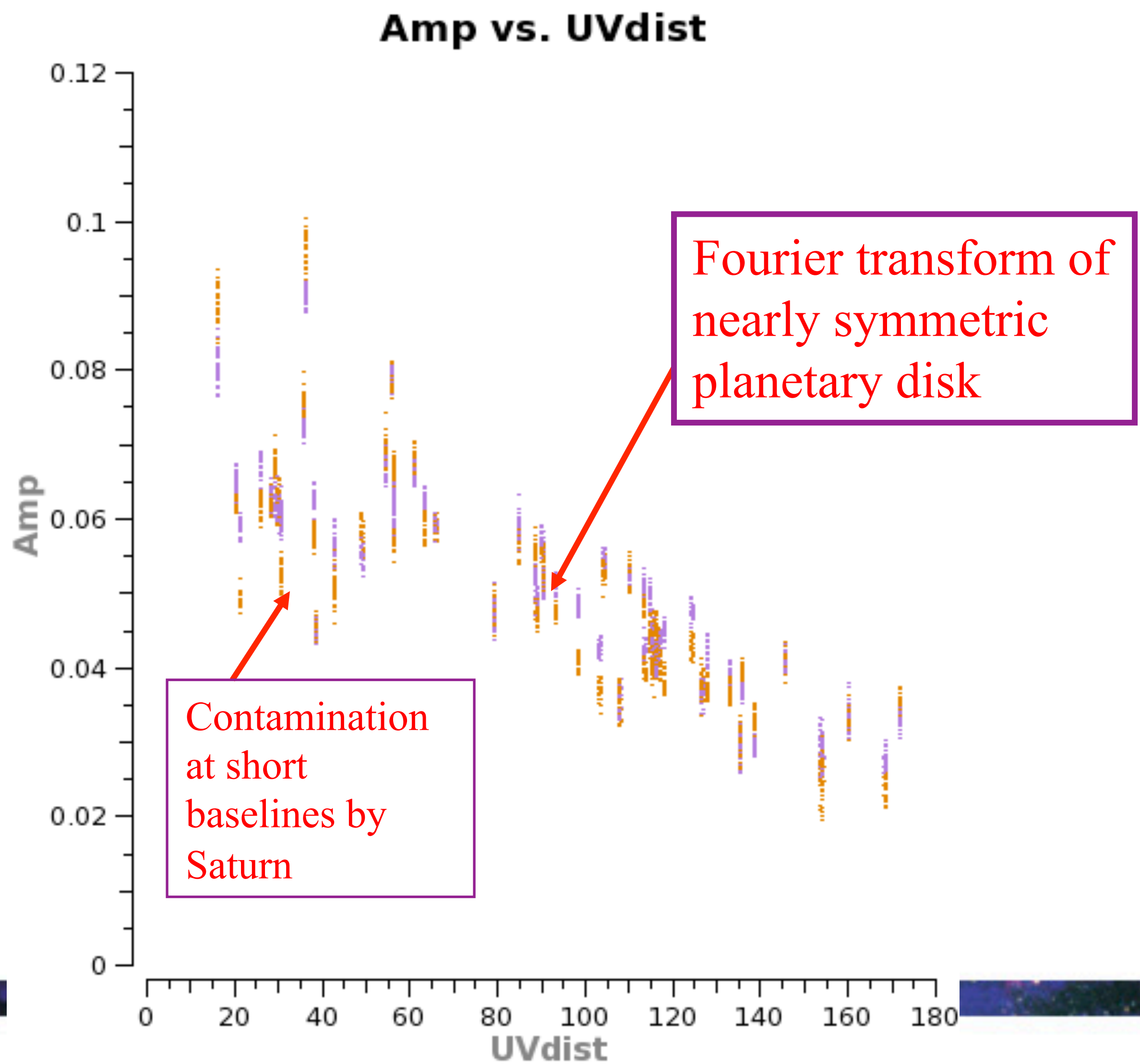
Real Example: VLA observes Jupiter

- A 6cm VLA observation of Jupiter:



Real Example: ALMA observes Titan

- From SV TWHydra casaguide



But

- 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

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

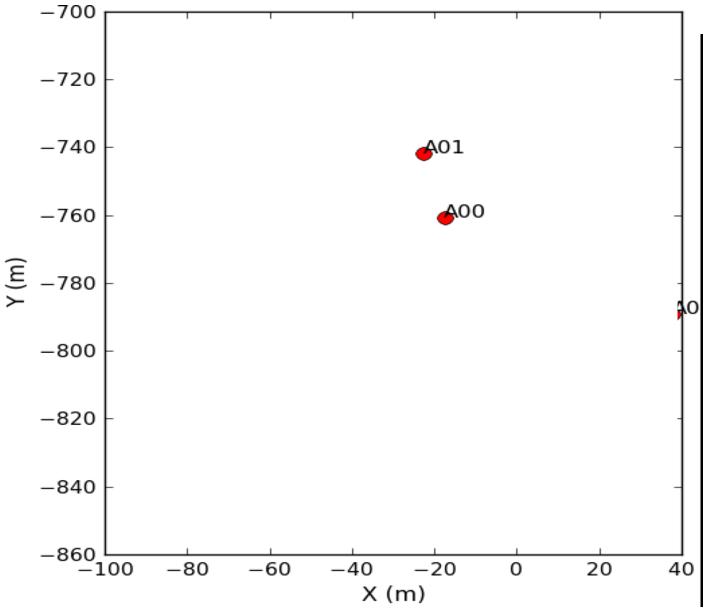
  where $b(x,y) = FT^{-1}\{B(u,v)\}$ (the point spread function)

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

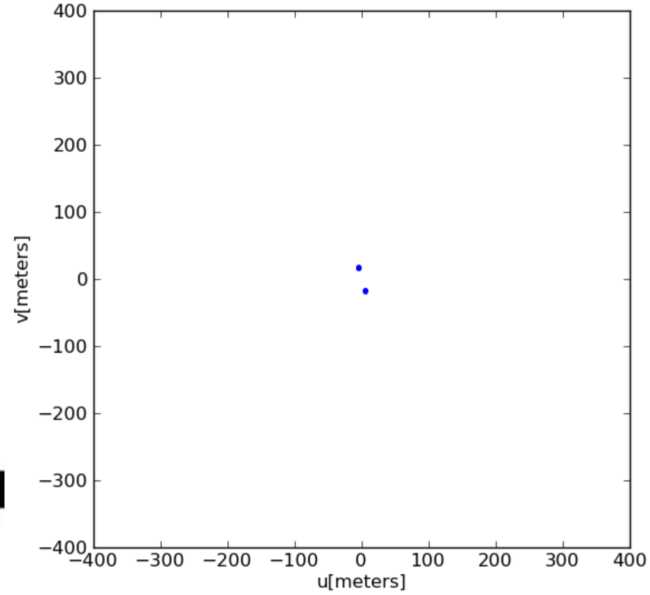
(the "dirty image" is the true image convolved with the "dirty beam")

NRAO

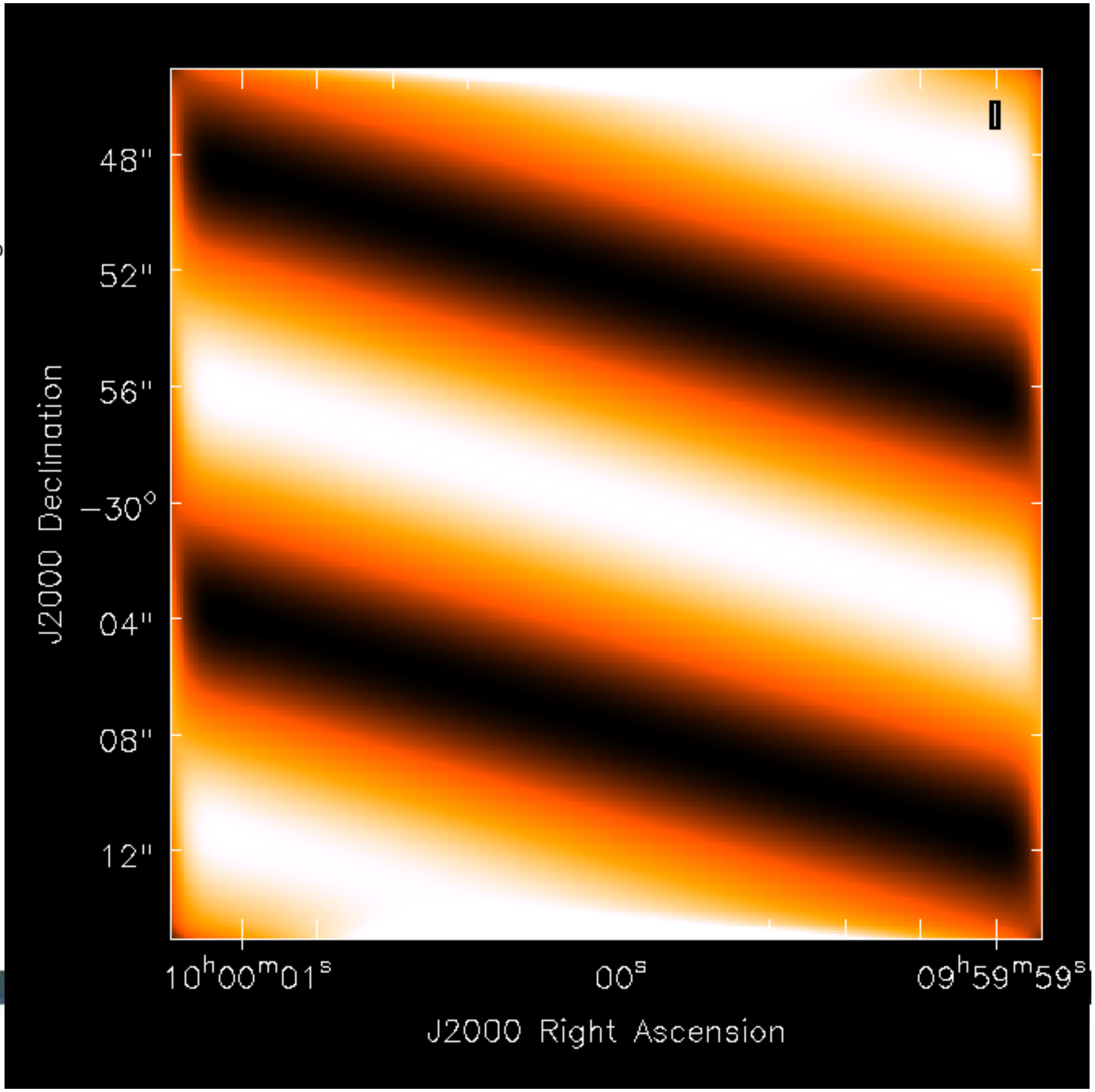
2 Antennas



A01
A00
Y (m)
X (m)

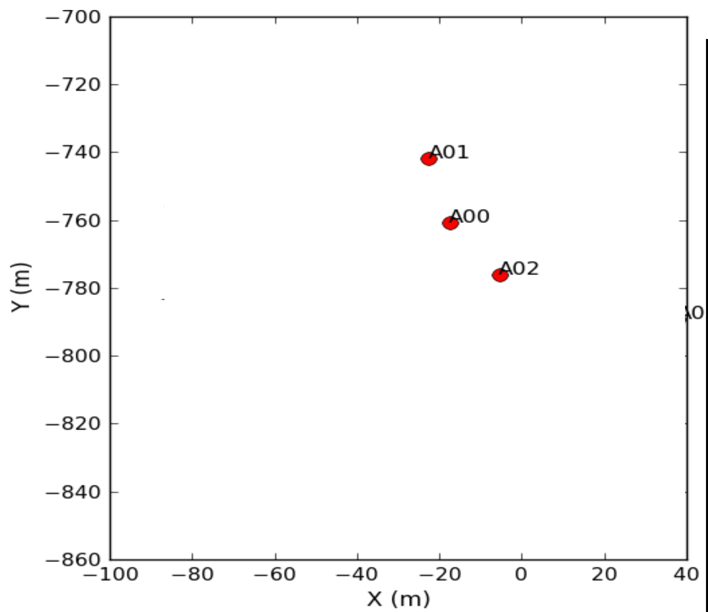


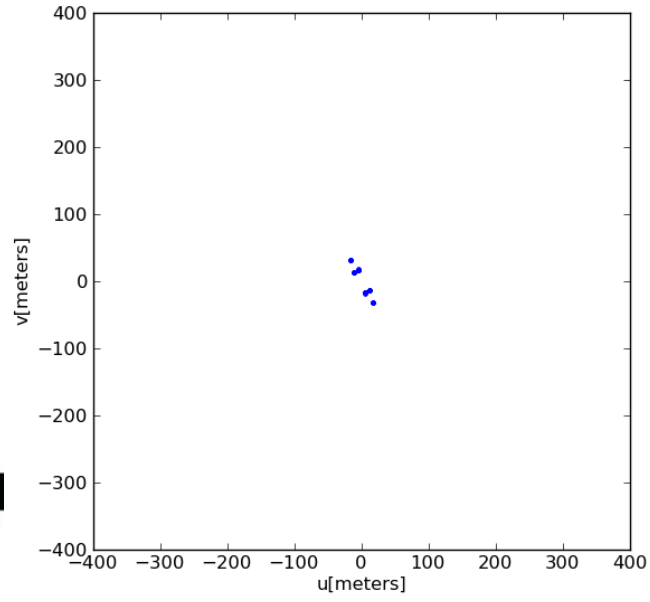
v[meters]
u[meters]

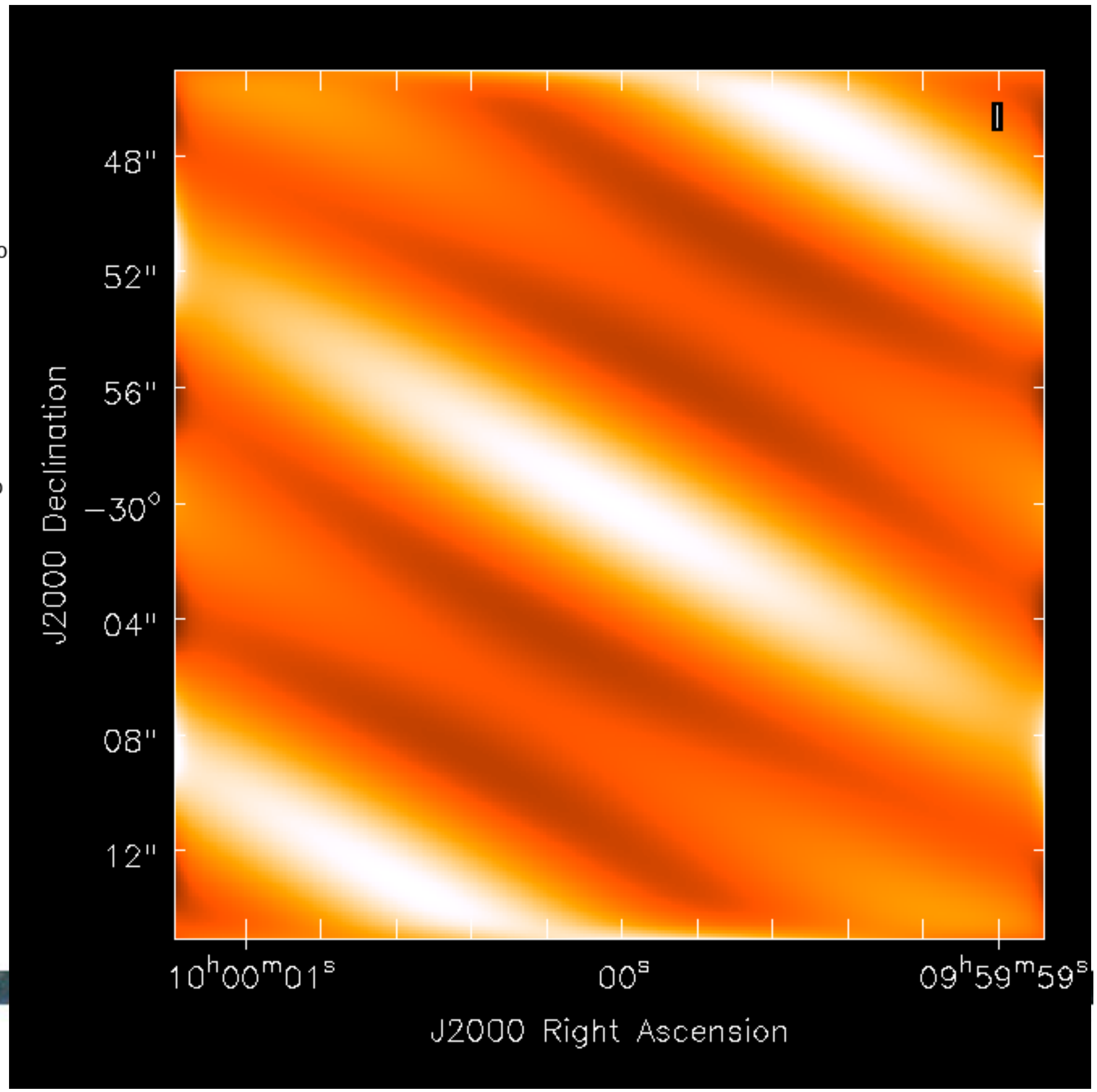


J2000 Declination
J2000 Right Ascension

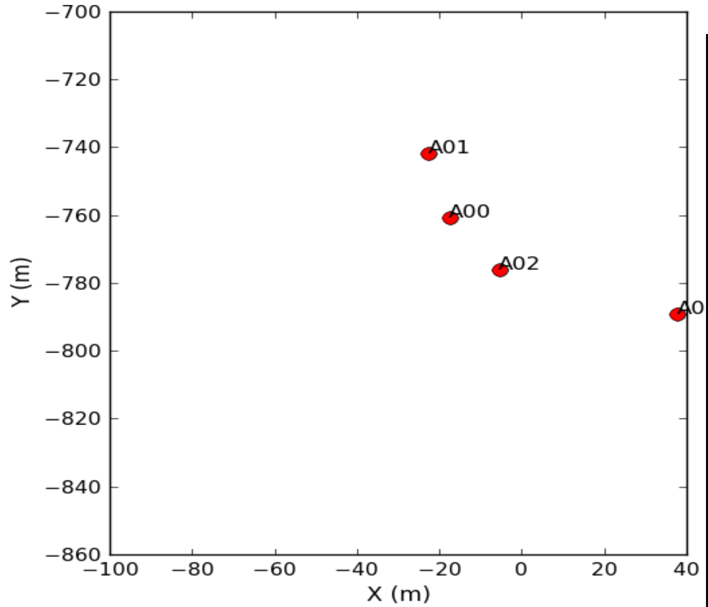
3 Antennas


A01
A00
A02
Y (m)
X (m)
−700
−720
−740
−760
−780
−800
−820
−840
−860
−100
−80
−60
−40
−20
0
20
40

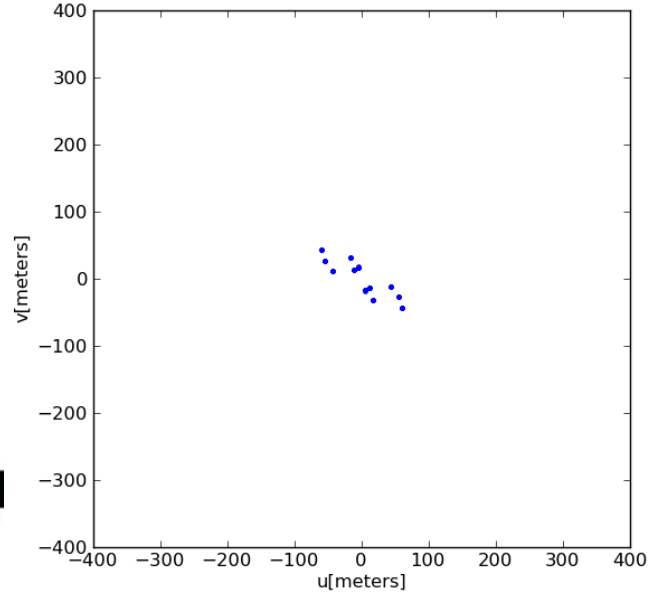

v[meters]
u[meters]
400
300
200
100
0
−100
−200
−300
−400


J2000 Declination
J2000 Right Ascension
48"
52"
56"
−30°
04"
08"
12"
10h00m01s
00s
09h59m59s
I

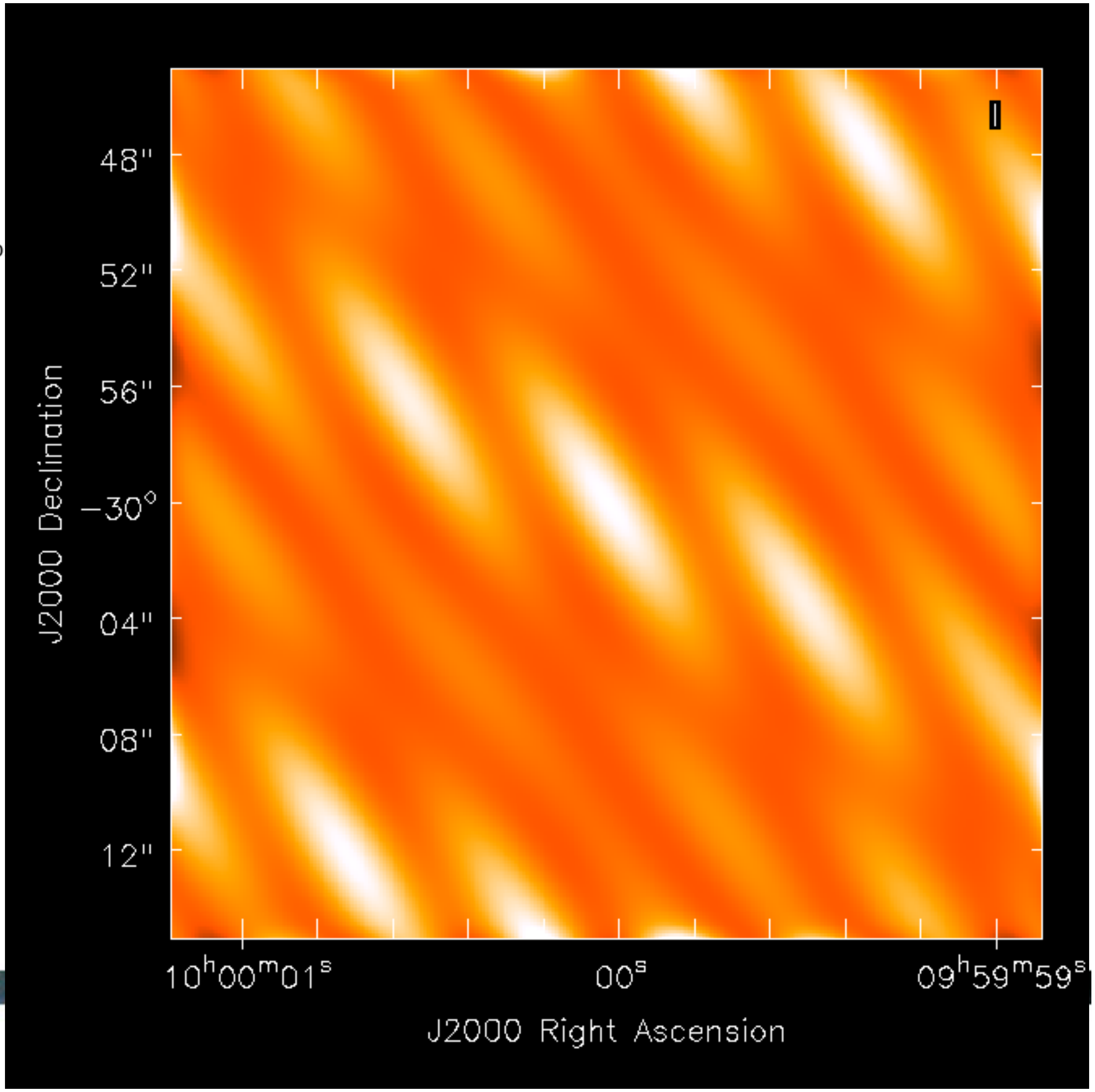
4 Antennas



A01
A00
A02
A0
Y (m)
X (m)
−700
−720
−740
−760
−780
−800
−820
−840
−860
−100
−80
−60
−40
−20
0
20
40

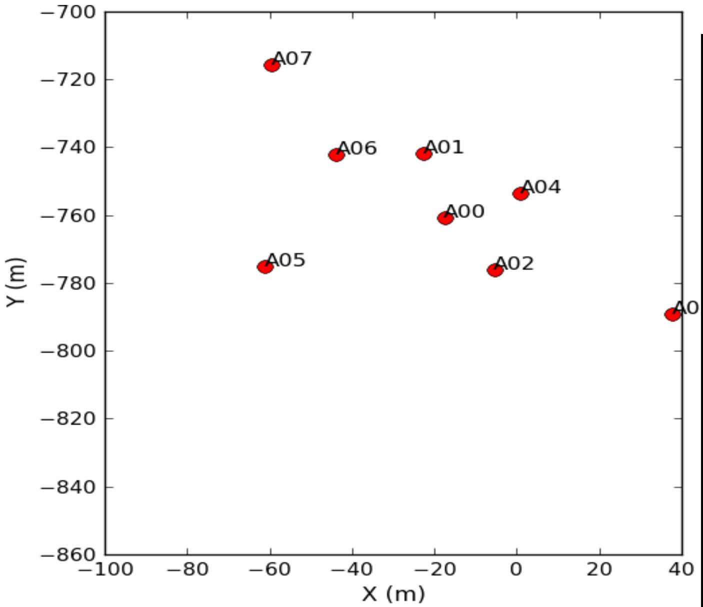


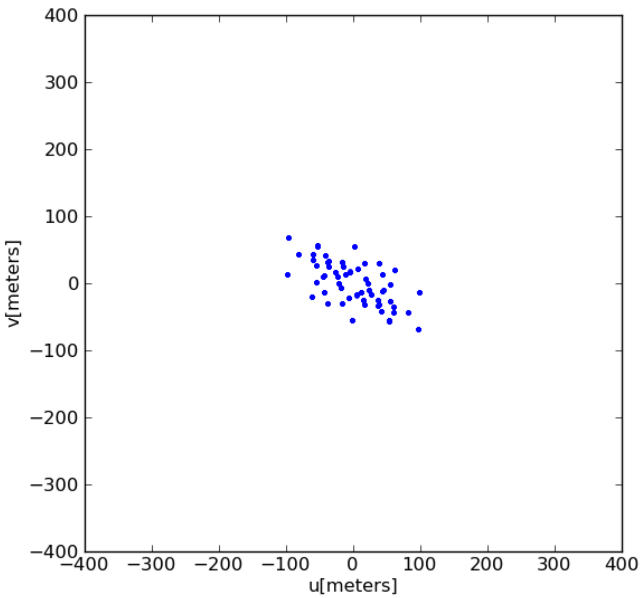
v[meters]
u[meters]
400
300
200
100
0
−100
−200
−300
−400
−400
−300
−200
−100
0
100
200
300
400

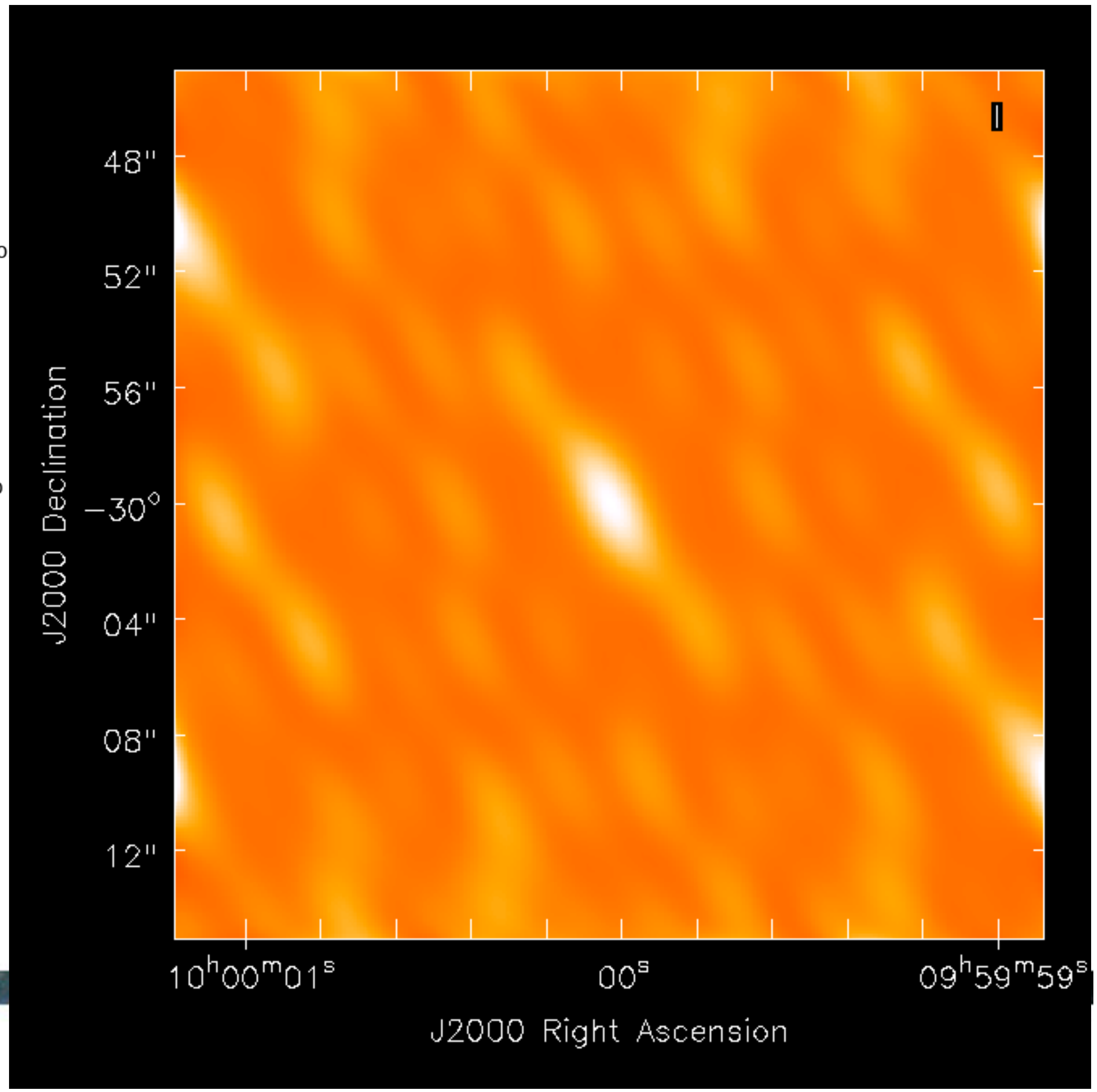


J2000 Declination
48"
52"
56"
−30°
04"
08"
12"
10h00m01s
00s
09h59m59s
J2000 Right Ascension

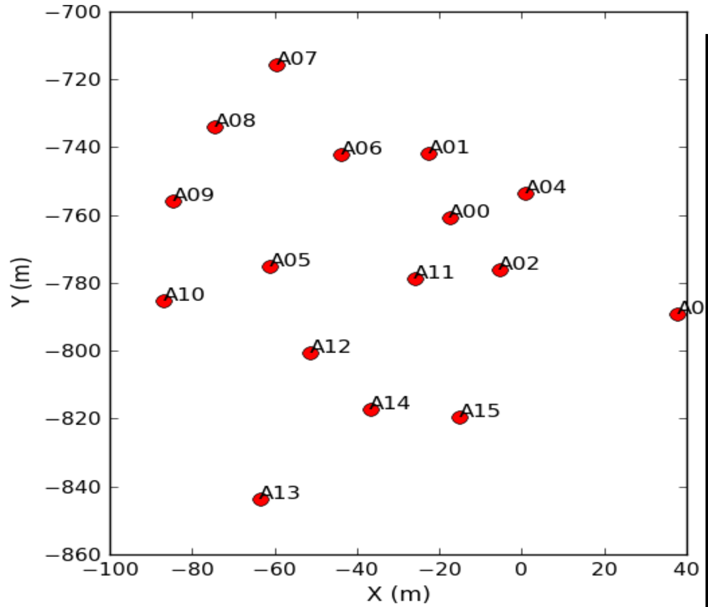
8 Antennas

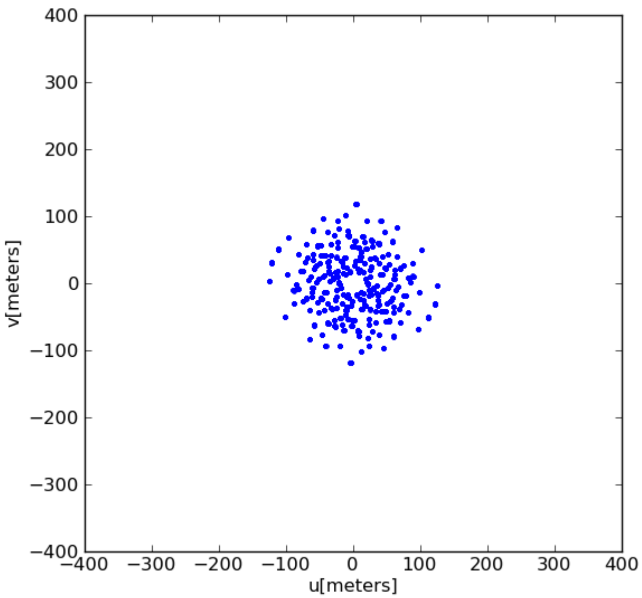

A07
A06
A01
A04
A00
A05
A02
Y (m)
X (m)
−700
−720
−740
−760
−780
−800
−820
−840
−860
−100 −80 −60 −40 −20 0 20 40

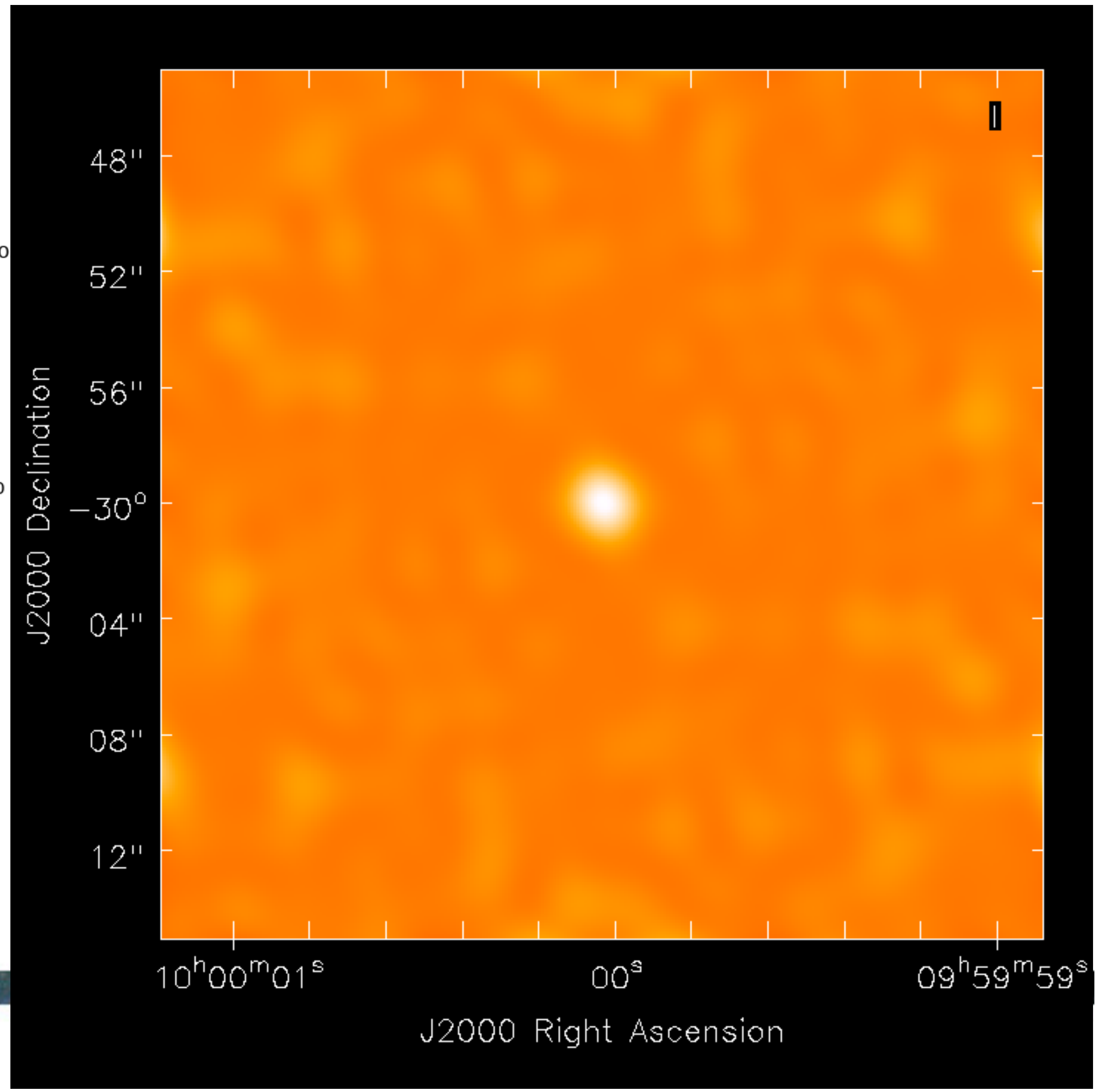

v[meters]
u[meters]
400
300
200
100
0
−100
−200
−300
−400
−400 −300 −200 −100 0 100 200 300 400


J2000 Declination
48"
52"
56"
−30°
04"
08"
12"
J2000 Right Ascension
10h00m01s
00s
09h59m59s

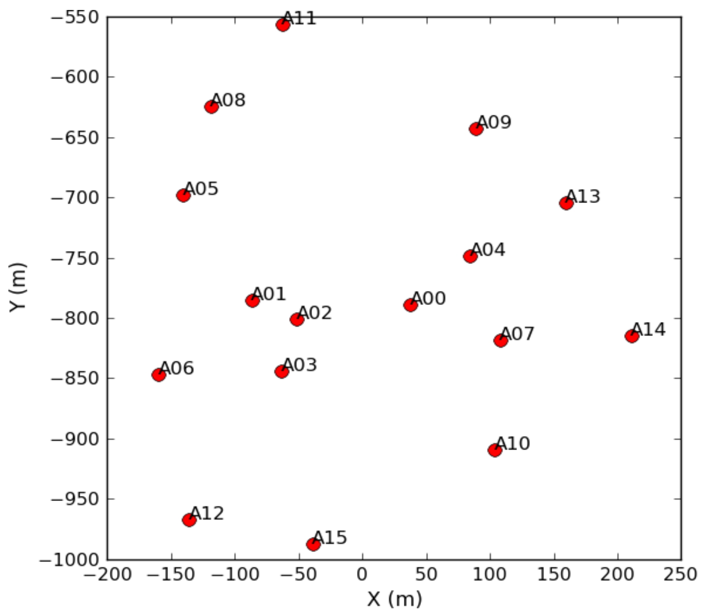
16 Antennas - Compact

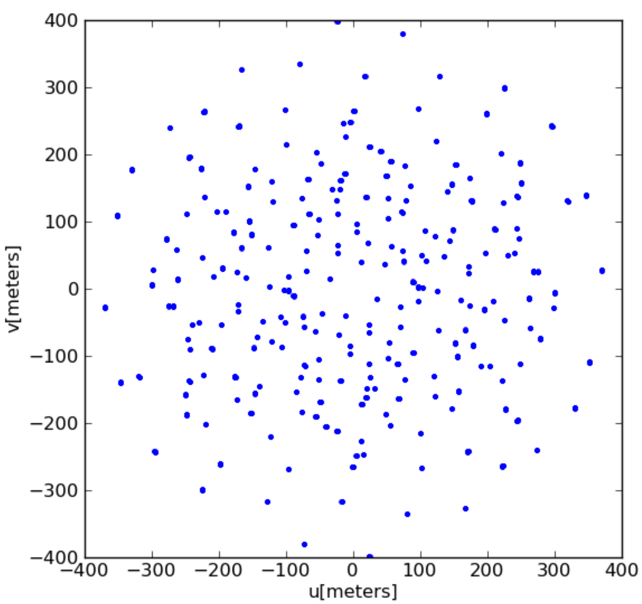


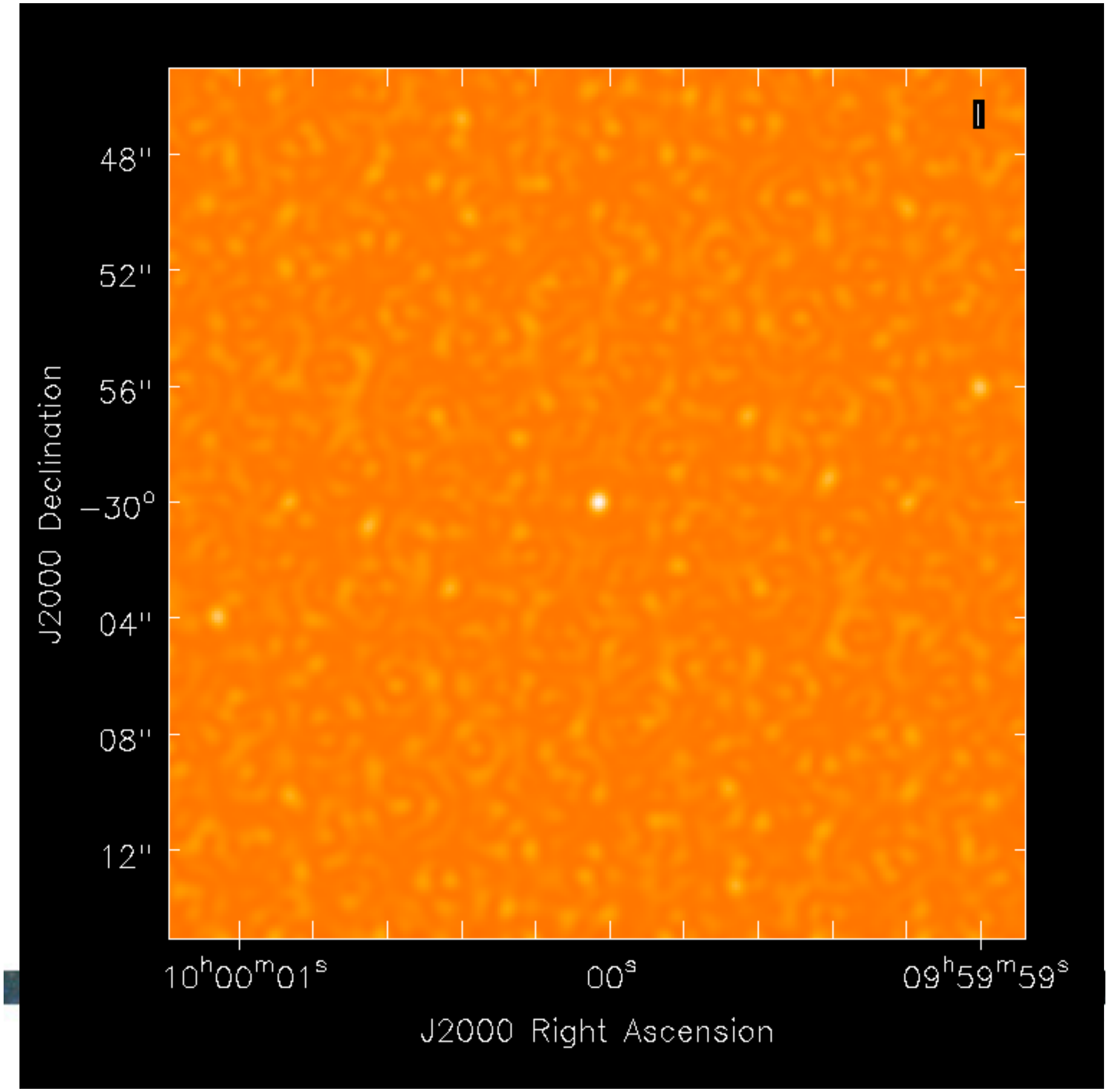




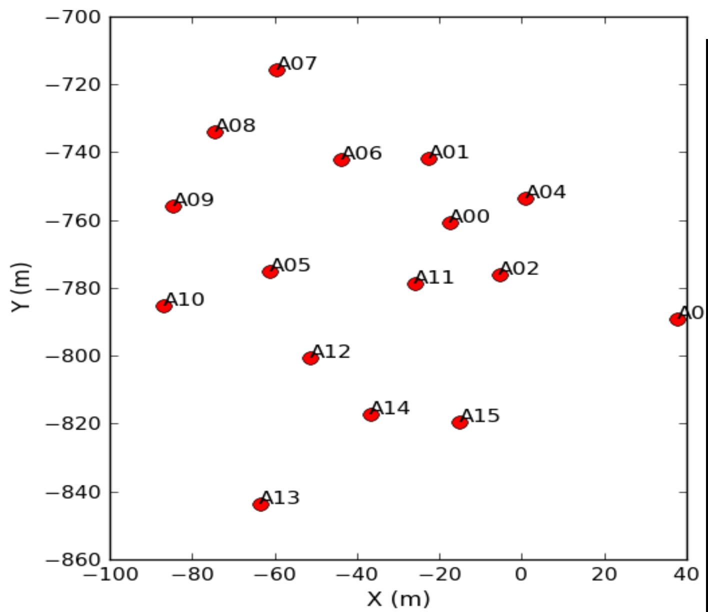
16 Antennas - Extended

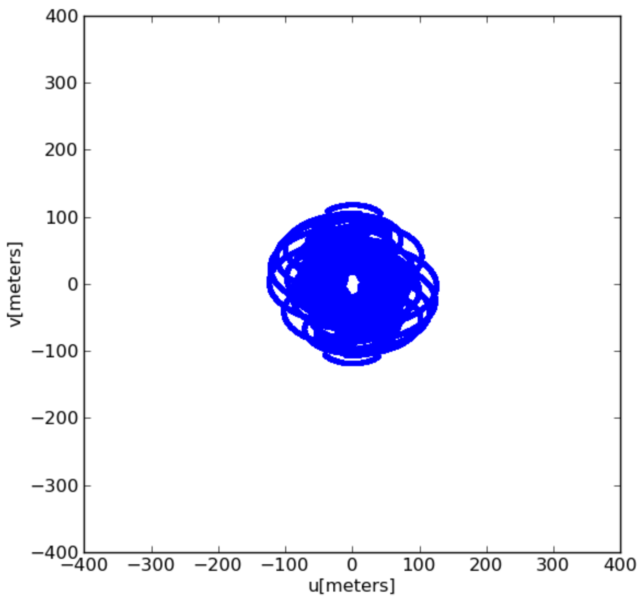


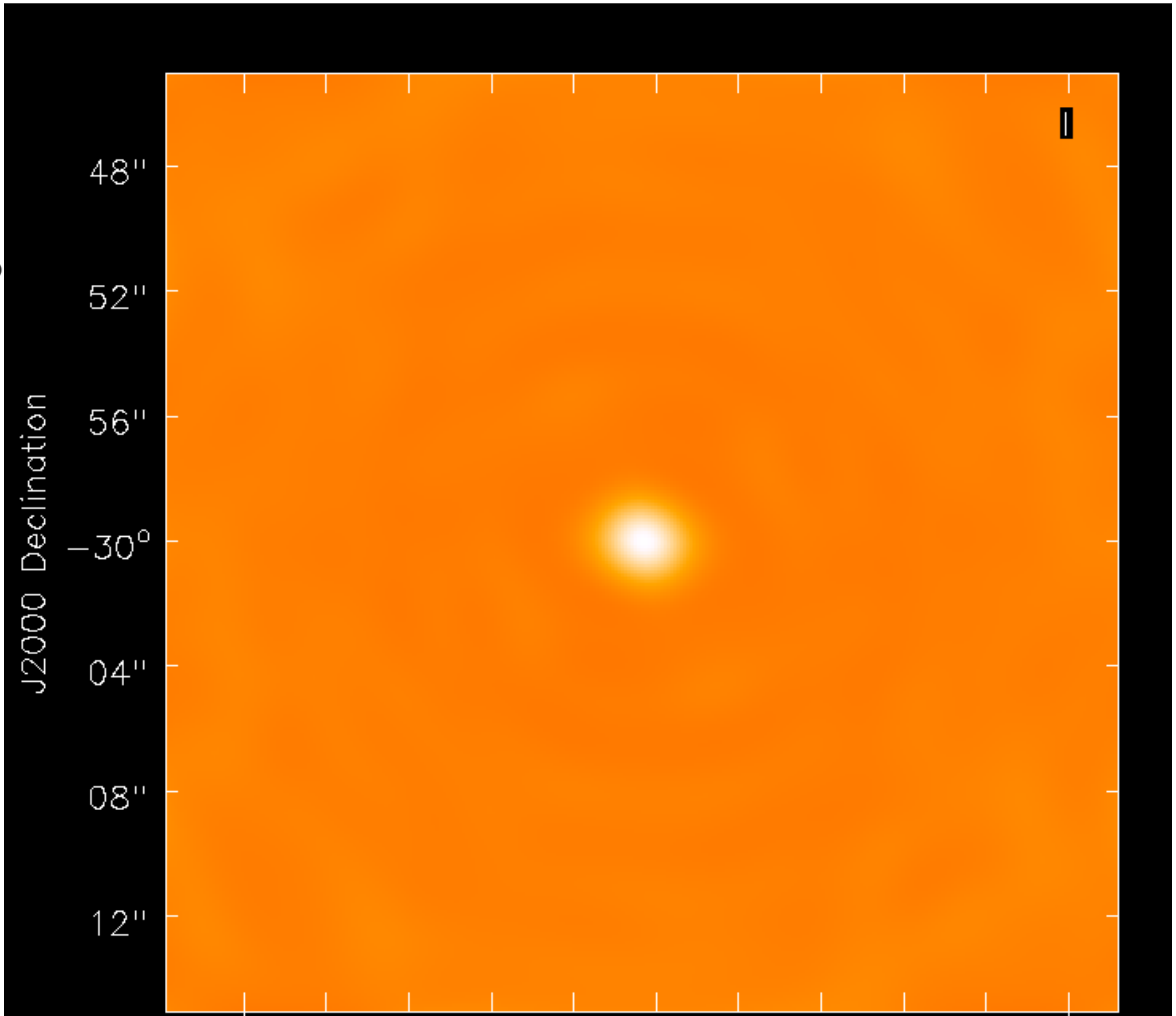




16 Antennas – Compact – 8 hours



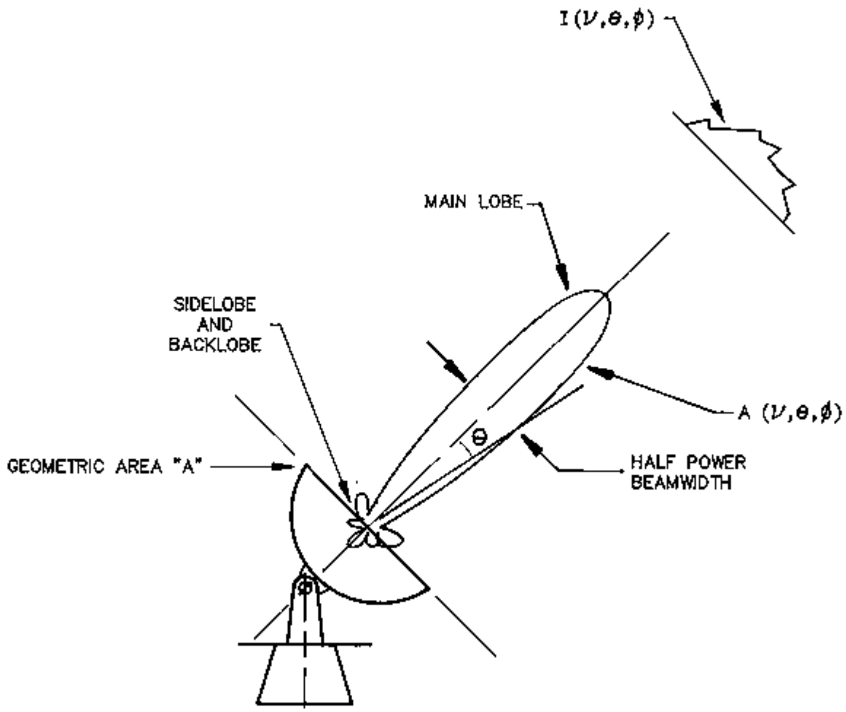




- Sky response = PSF = dirty beam = Fourier transform of uv distribution

(imperfect) Reconstruction of the Sky

- Incomplete sampling of uv plane: sidelobes
- non-point-like instrumental response ***A***   a.k.a finite primary beam



$$V(u,v) = \boldsymbol{A} \otimes FT\{\, T(x,y) \,\}$$

(imperfect) Reconstruction of the Sky

- Incomplete sampling of uv plane: sidelobes
- non-point-like instrumental response ***A***
- noise

$$V(u,v) = \boldsymbol{A} \otimes FT\{\ T(x,y)\ \} + n$$

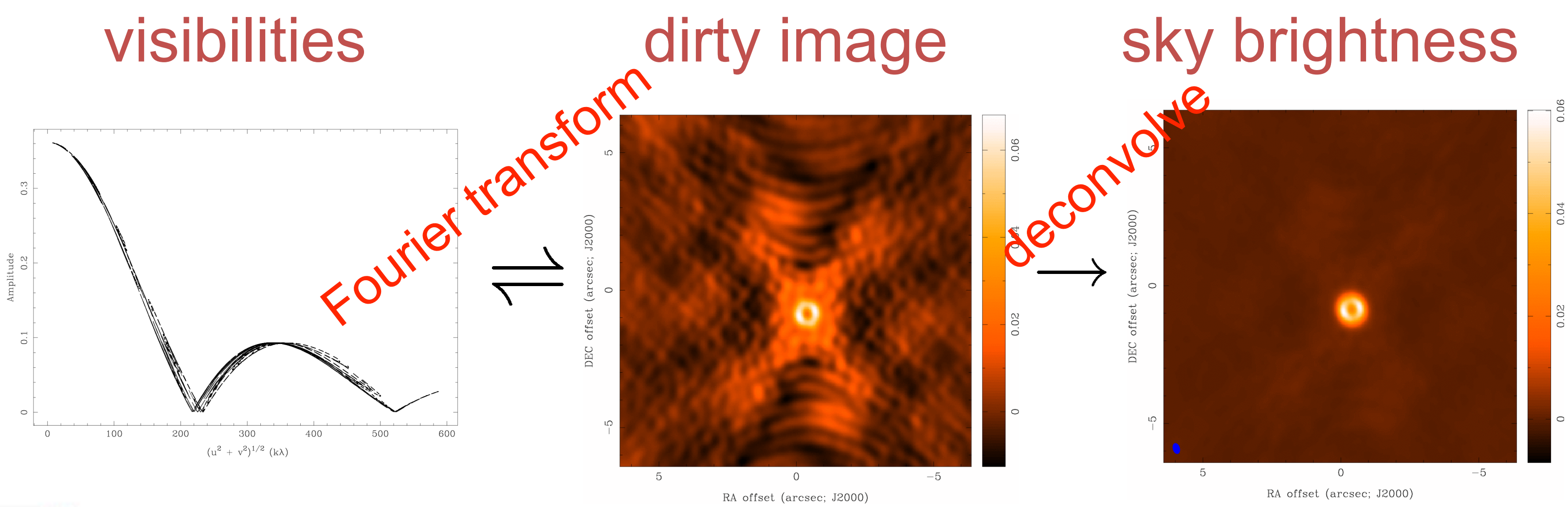
- Need to invert, but
  - FFT requires information on a regular grid in uv plane

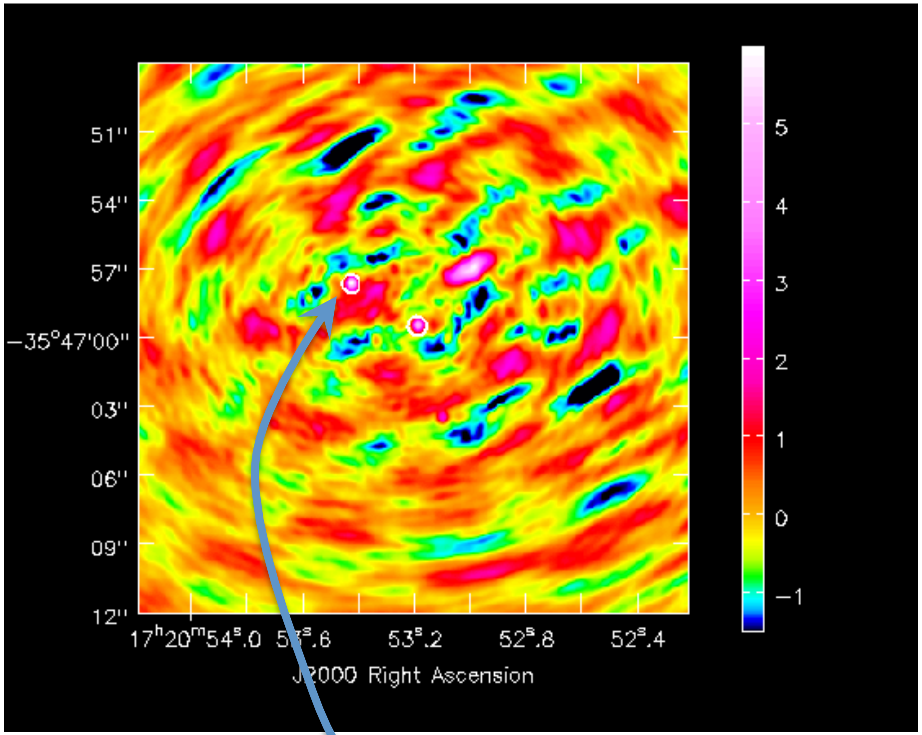
$$V^G(u,v) = V(u,v)B(u,v) \otimes G(u,v) \rightleftharpoons T^D(x,y)g(x,y)$$

  - Ideal: gridding would invert instrument response ***A*** = FT of primary beam, but its not invertible.
  - Use spheroidal function for single-field imaging, use PB for mosaic imaging
    * spheroidal are generally more tapered, less aliasing

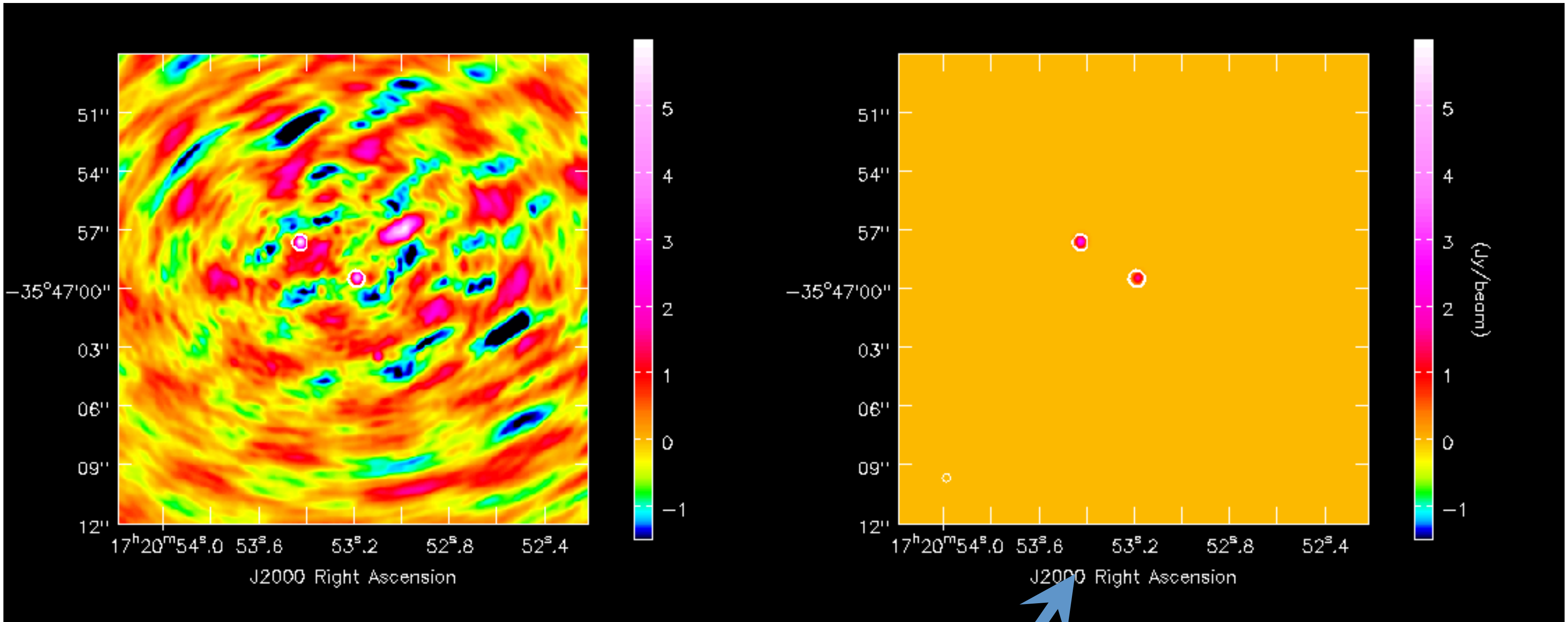
How to analyze (imperfect) interferometer data?

- uv plane analysis
  - best for “simple” sources, e.g. point sources, disks
- 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) $T(x,y)$

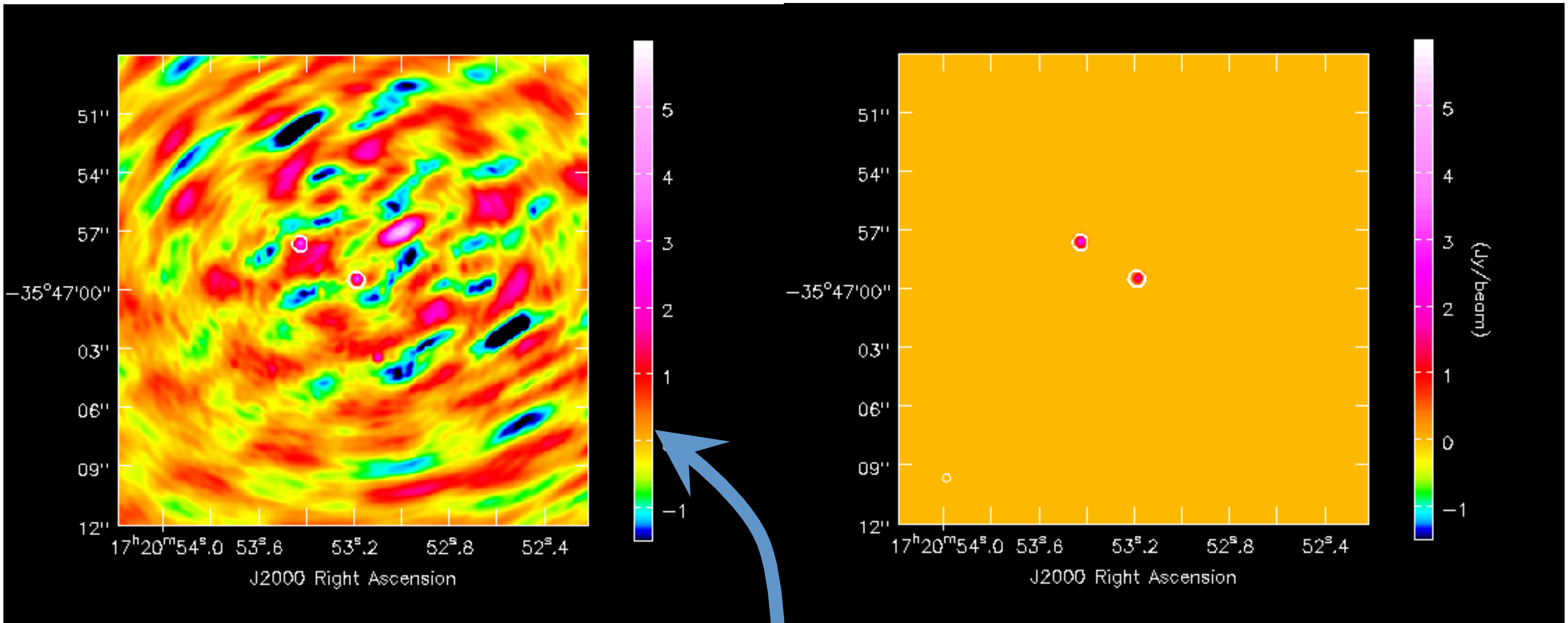




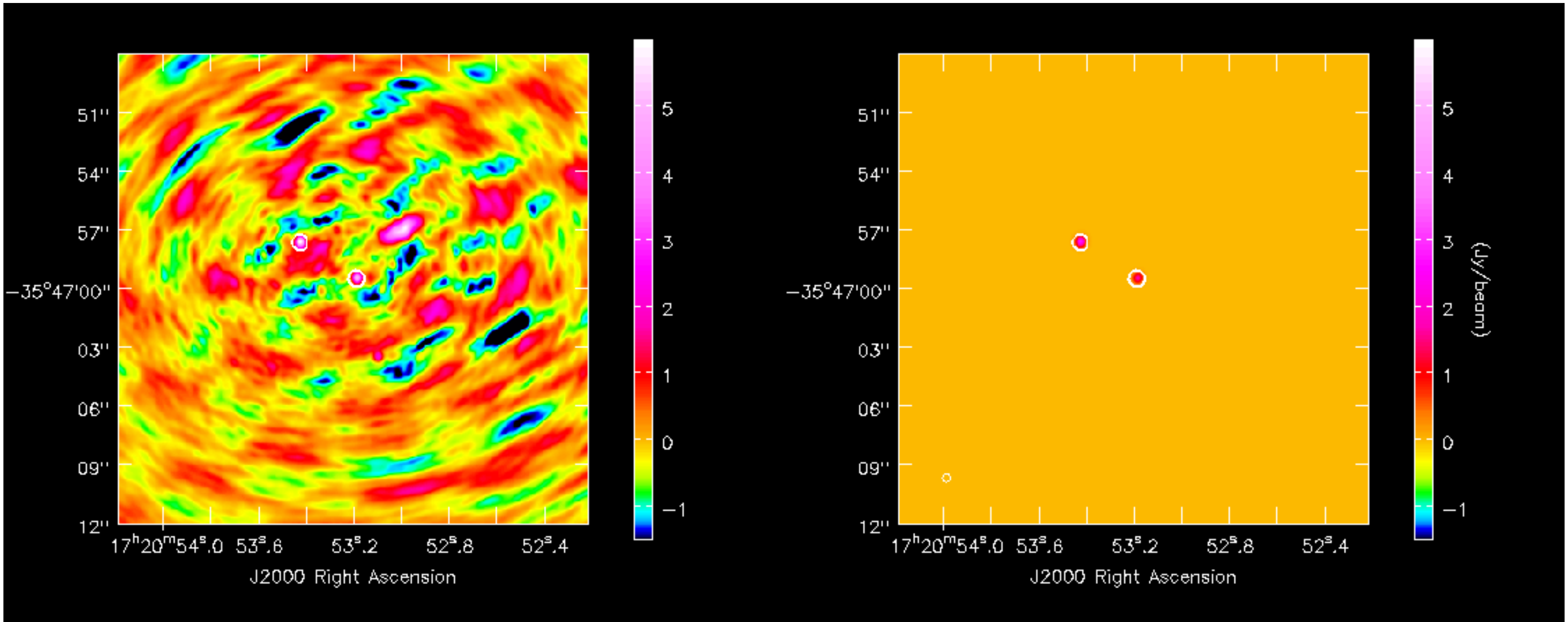
- Find brightest points in dirty image



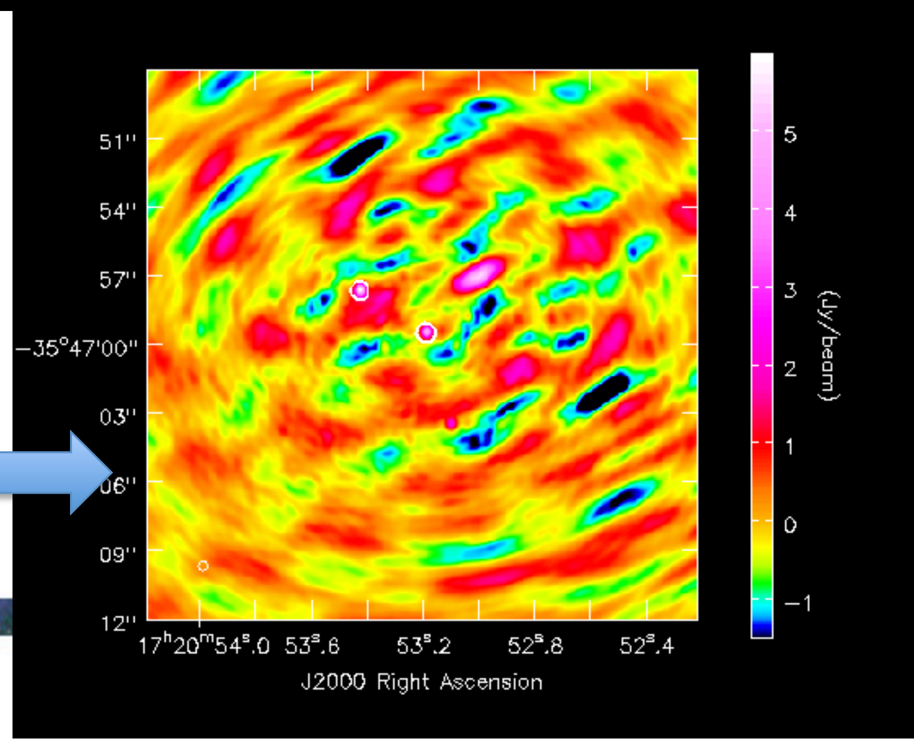
- Find brightest points in dirty image
- Create model image containing a fraction of those flux points

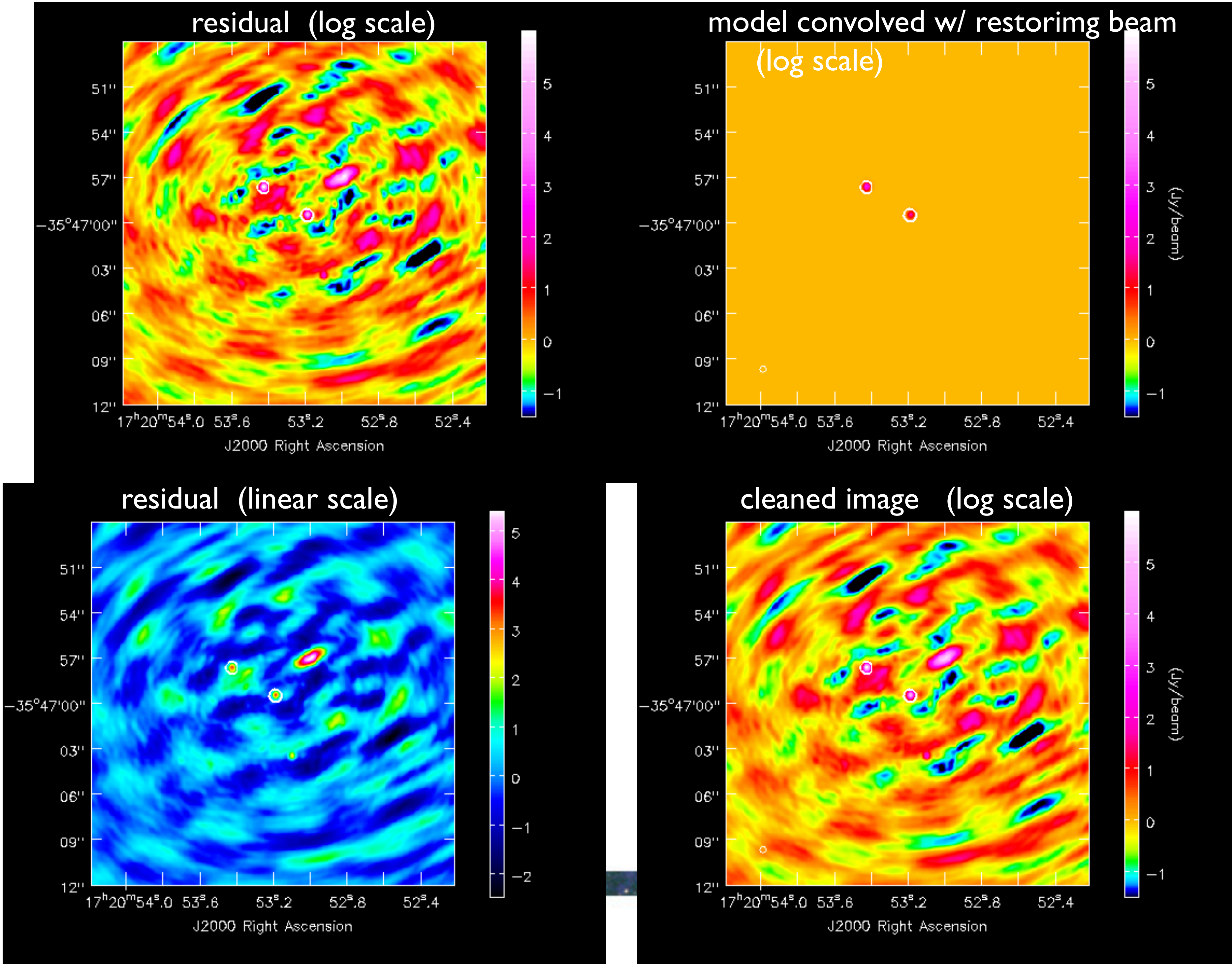


- Find brightest points in dirty image
- Create model image containing a fraction of those flux points
- Subtract model from data, leaving a residual

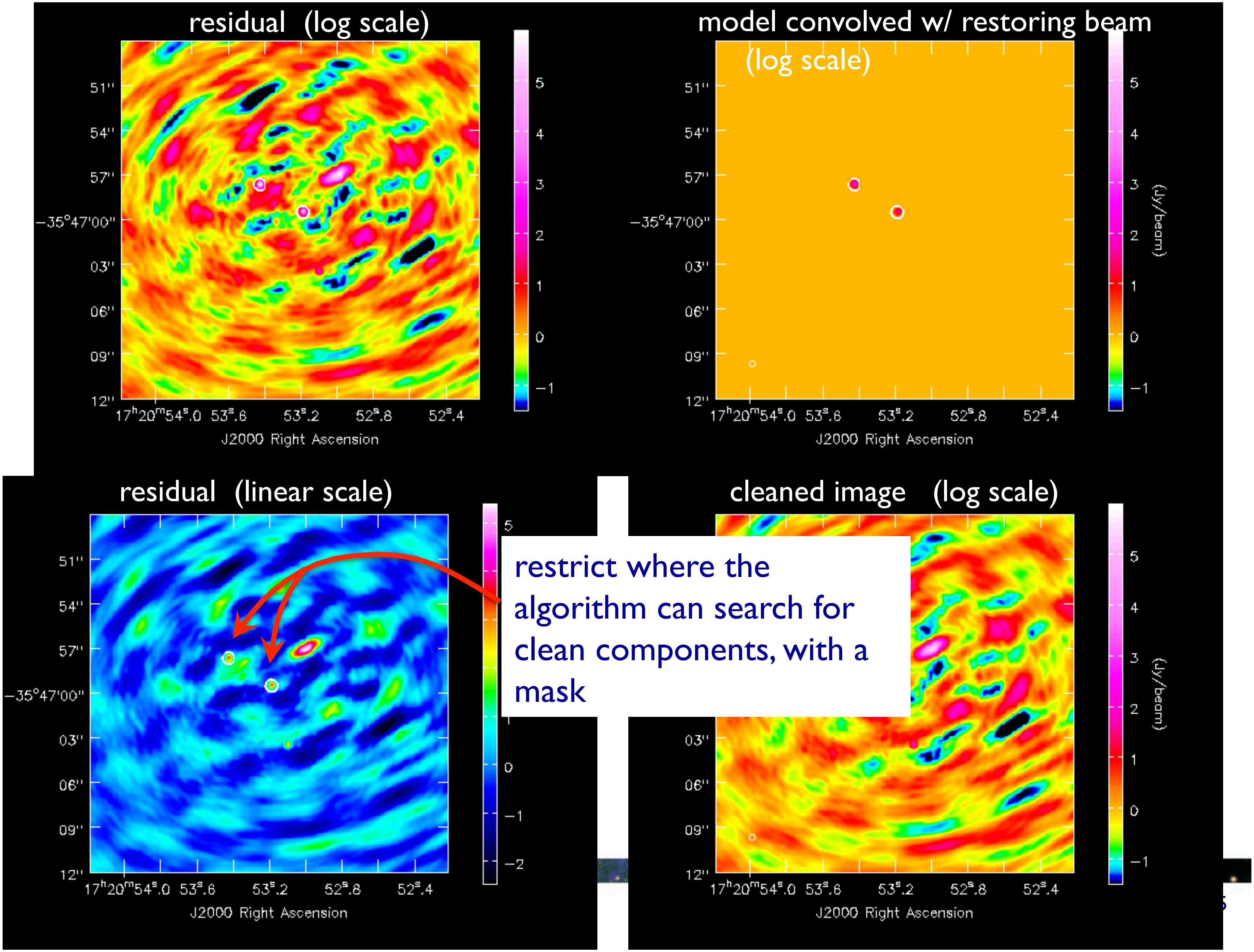


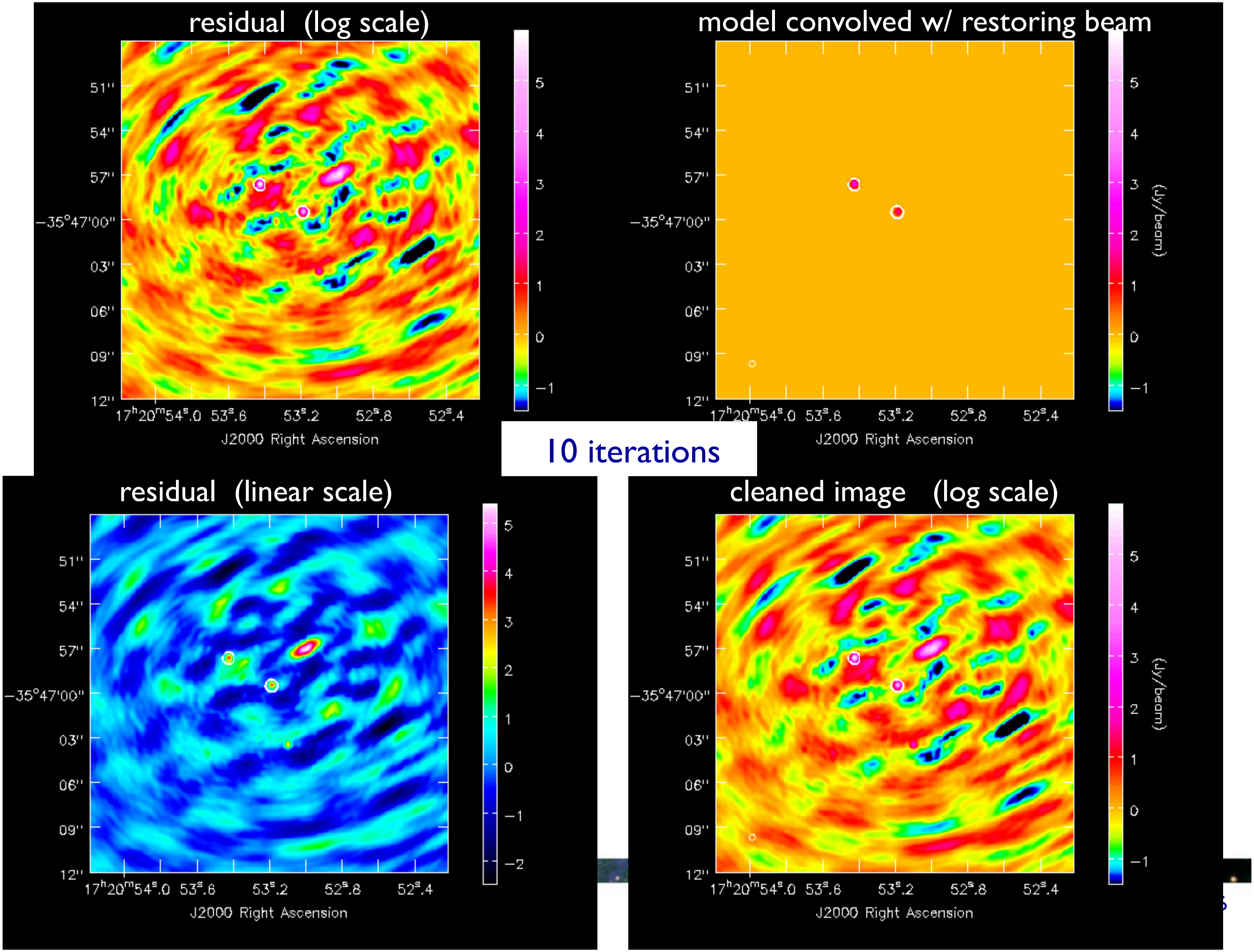
- Find brightest points in dirty image
- Create model image containing a fraction of those flux points
- Subtract model from data, leaving a residual
- Final product = residual + model (convolved with restoring Gaussian beam)





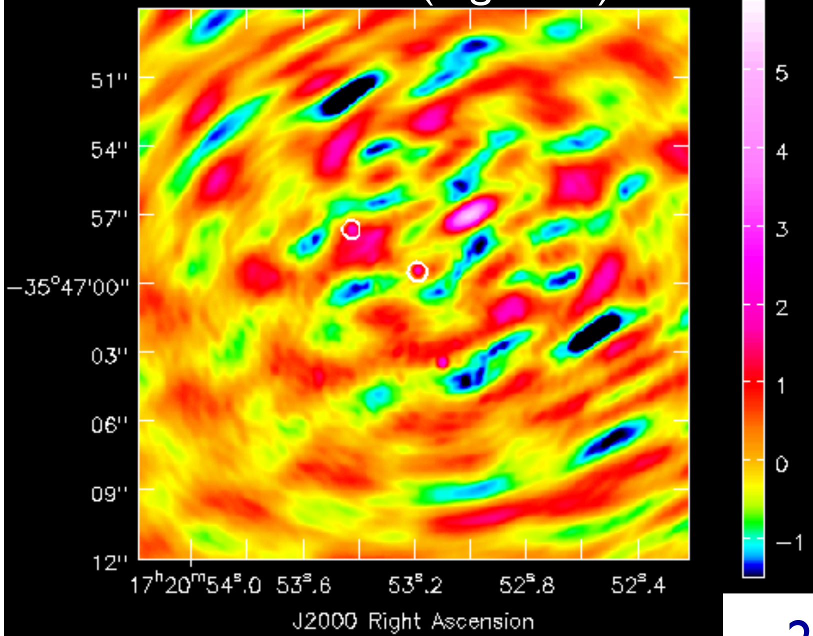
residual (log scale)
model convolved w/ restorimg beam (log scale)
residual (linear scale)
cleaned image (log scale)
51''
54''
57''
−35°47'00''
03''
06''
09''
12''
17h20m54s.0 53s.6 53s.2 52s.8 52s.4
J2000 Right Ascension
(Jy/beam)
5
4
3
2
1
0
−1
−2


residual (log scale)
model convolved w/ restoring beam (log scale)
residual (linear scale)
cleaned image (log scale)
restrict where the algorithm can search for clean components, with a mask
J2000 Right Ascension
17h20m54s.0 53s.6 53s.2 52s.8 52s.4
51''
54''
57''
−35°47'00''
03''
06''
09''
12''
(Jy/beam)
5
4
3
2
1
0
−1
−2

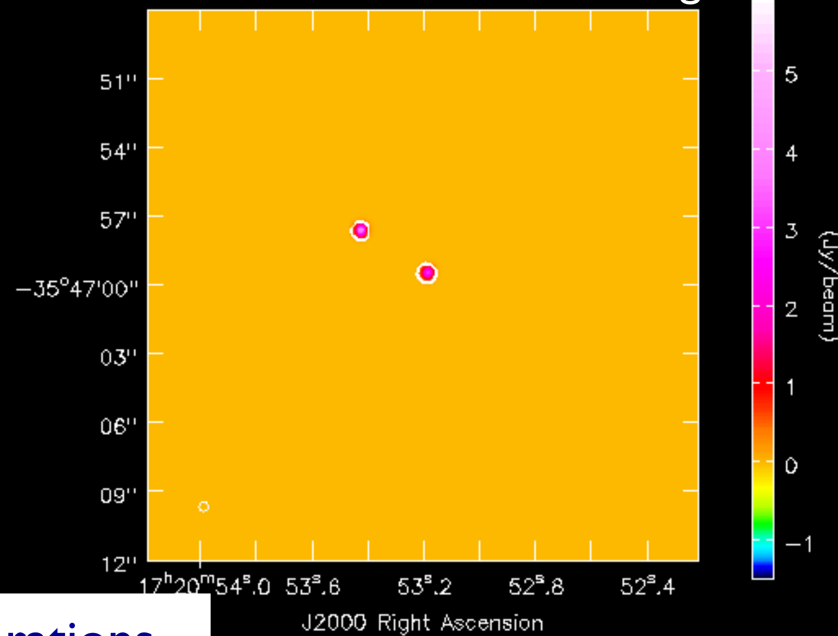


10 iterations

residual (log scale)

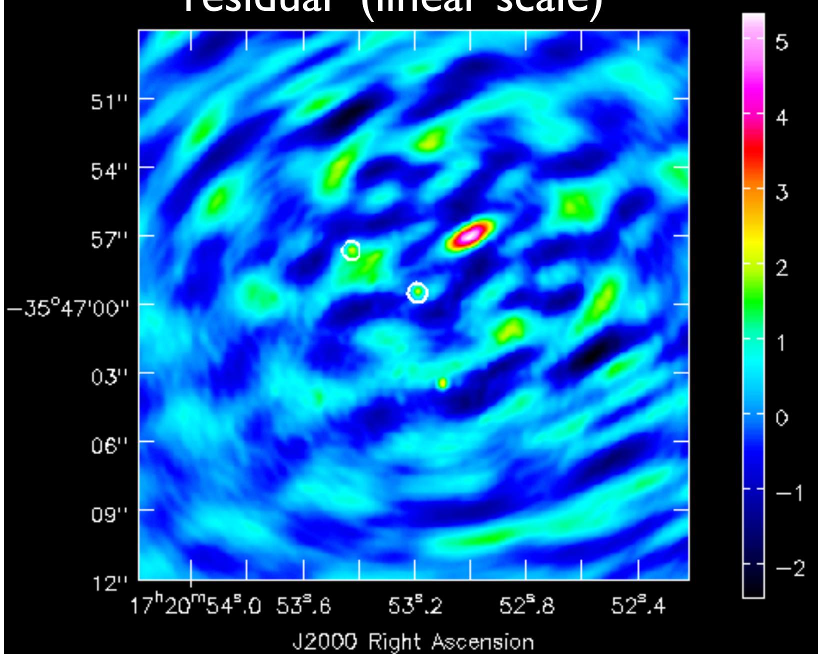


model convolved w/ restoring beam

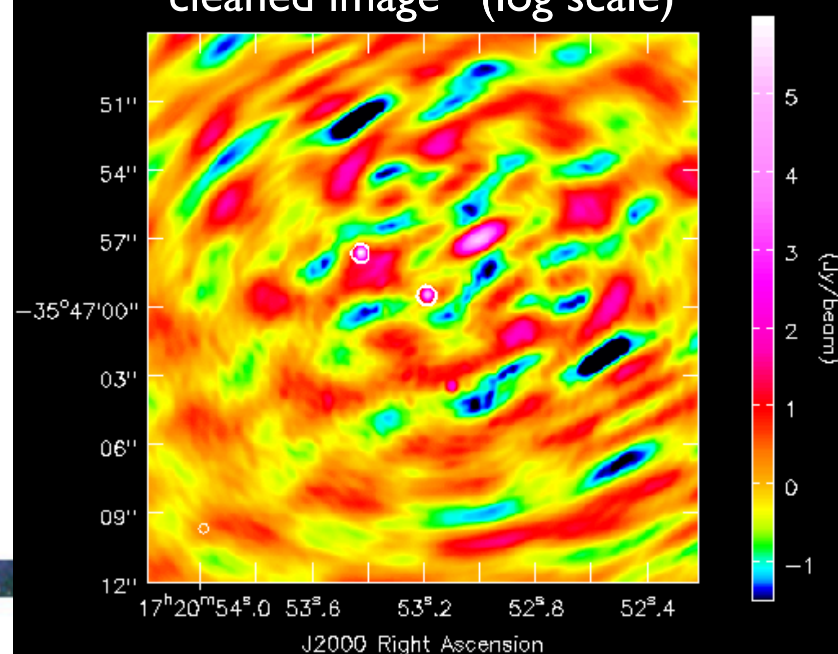


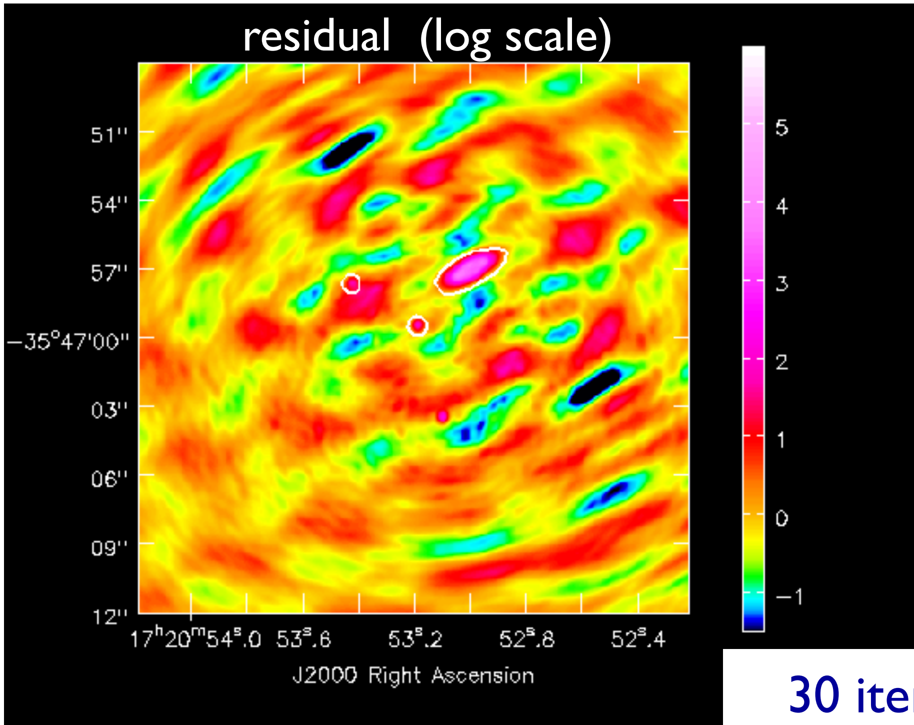
20 iterations

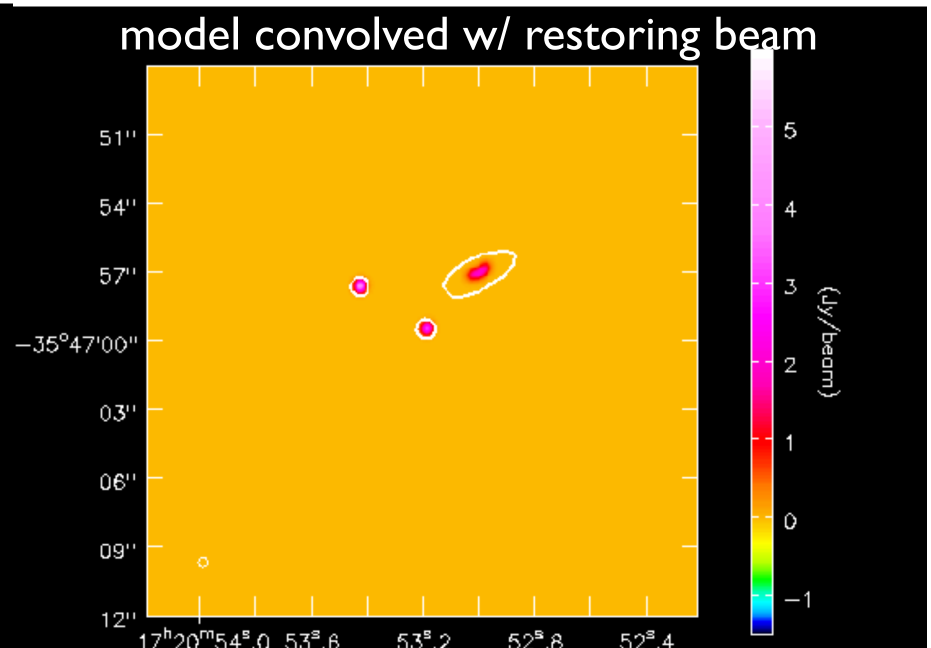
residual (linear scale)



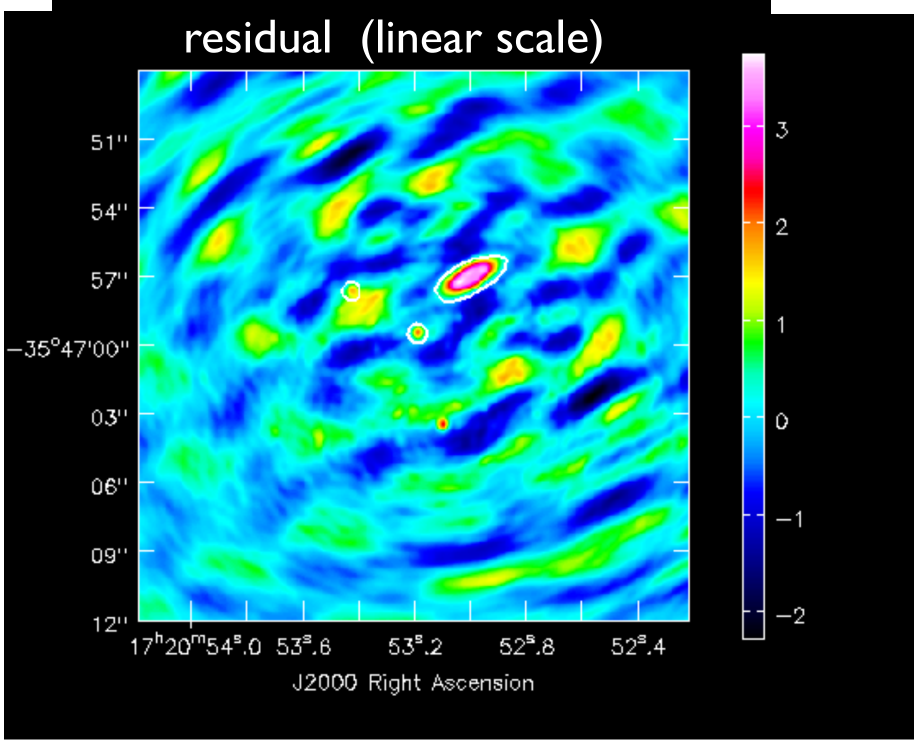
cleaned image (log scale)

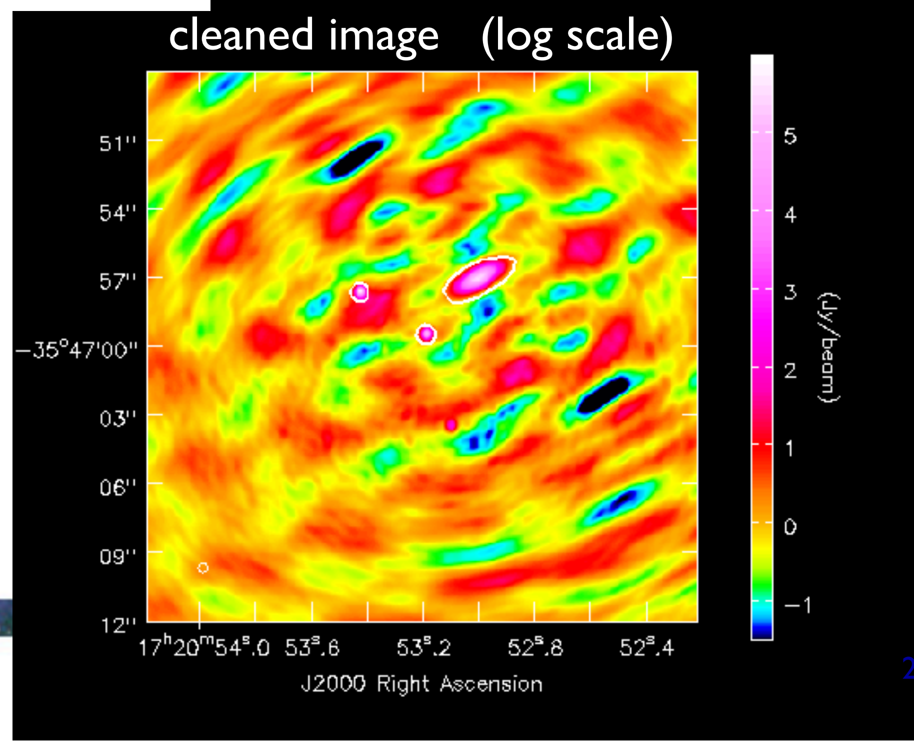


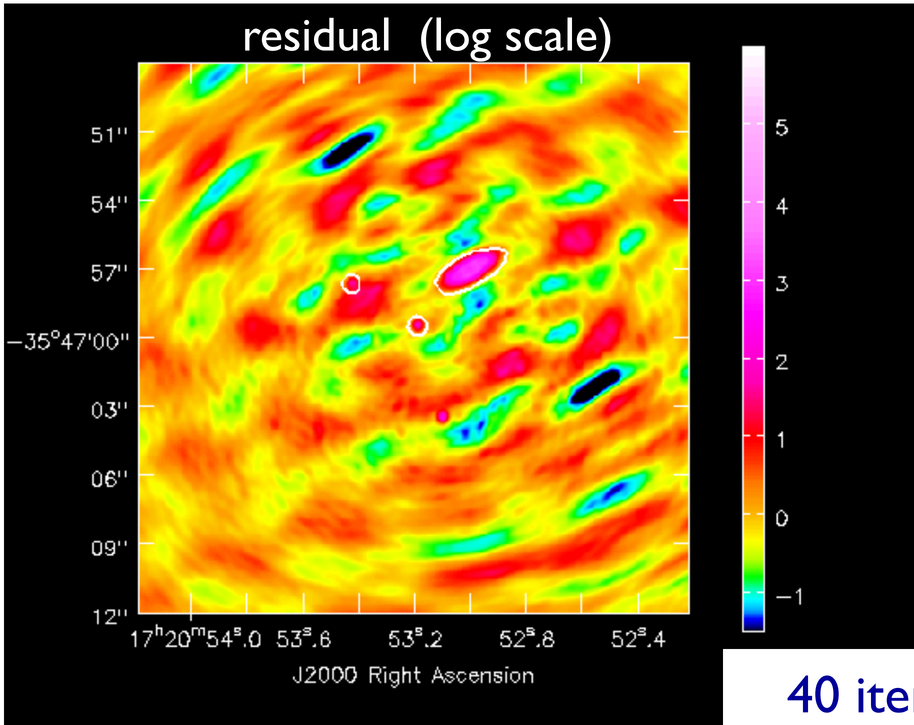


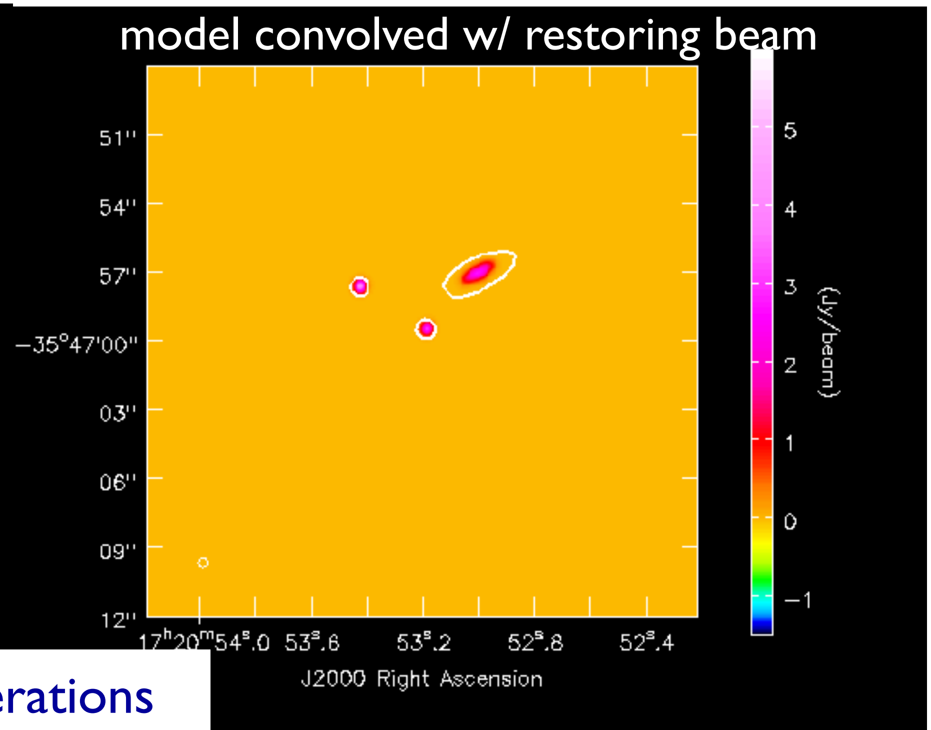


30 iterations

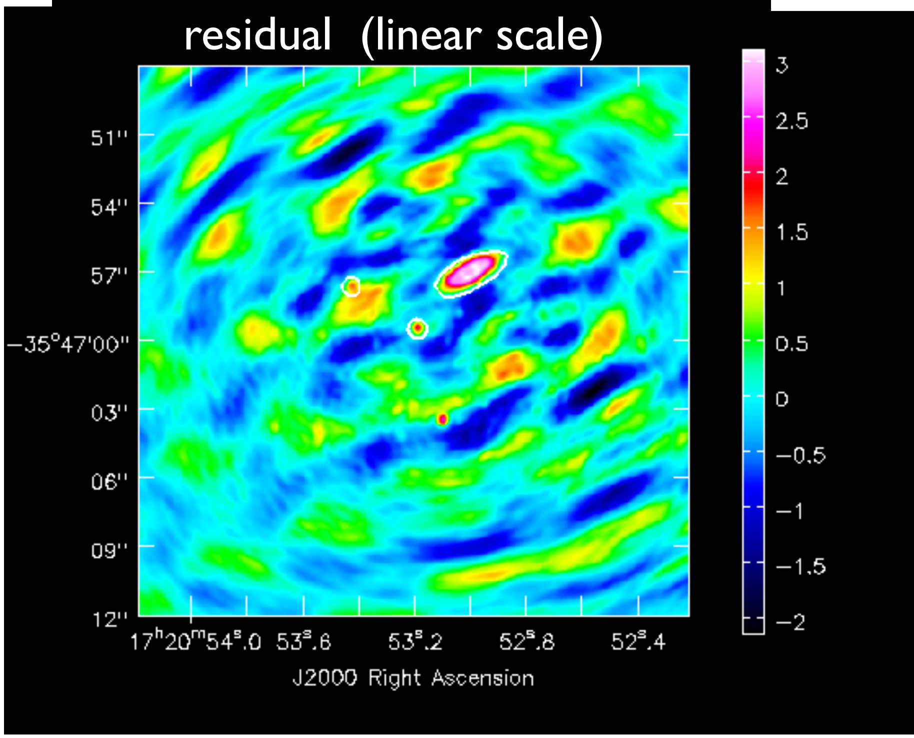


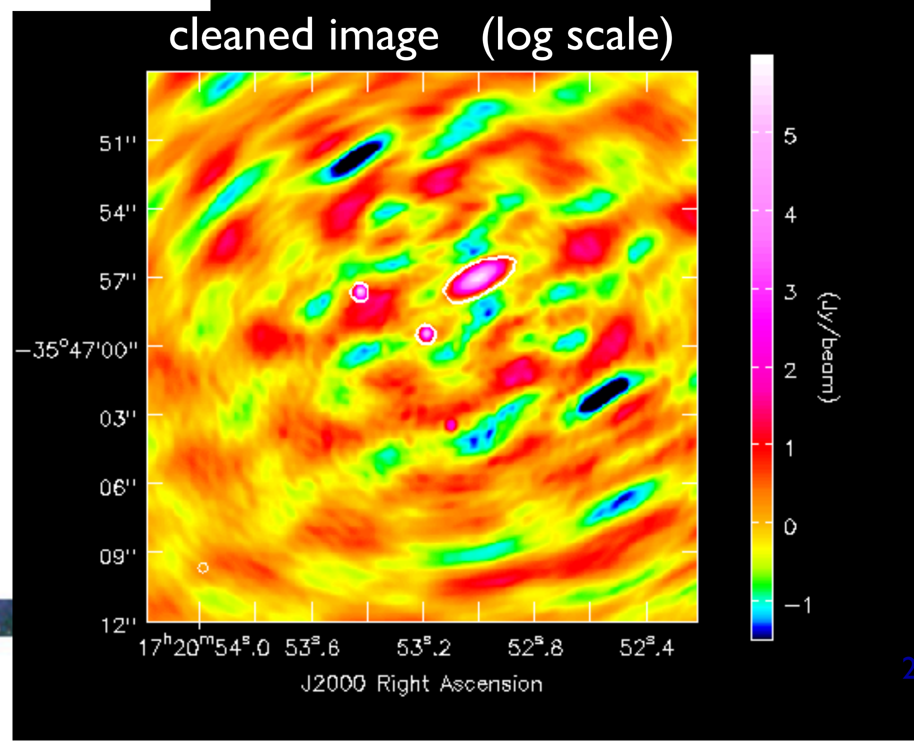


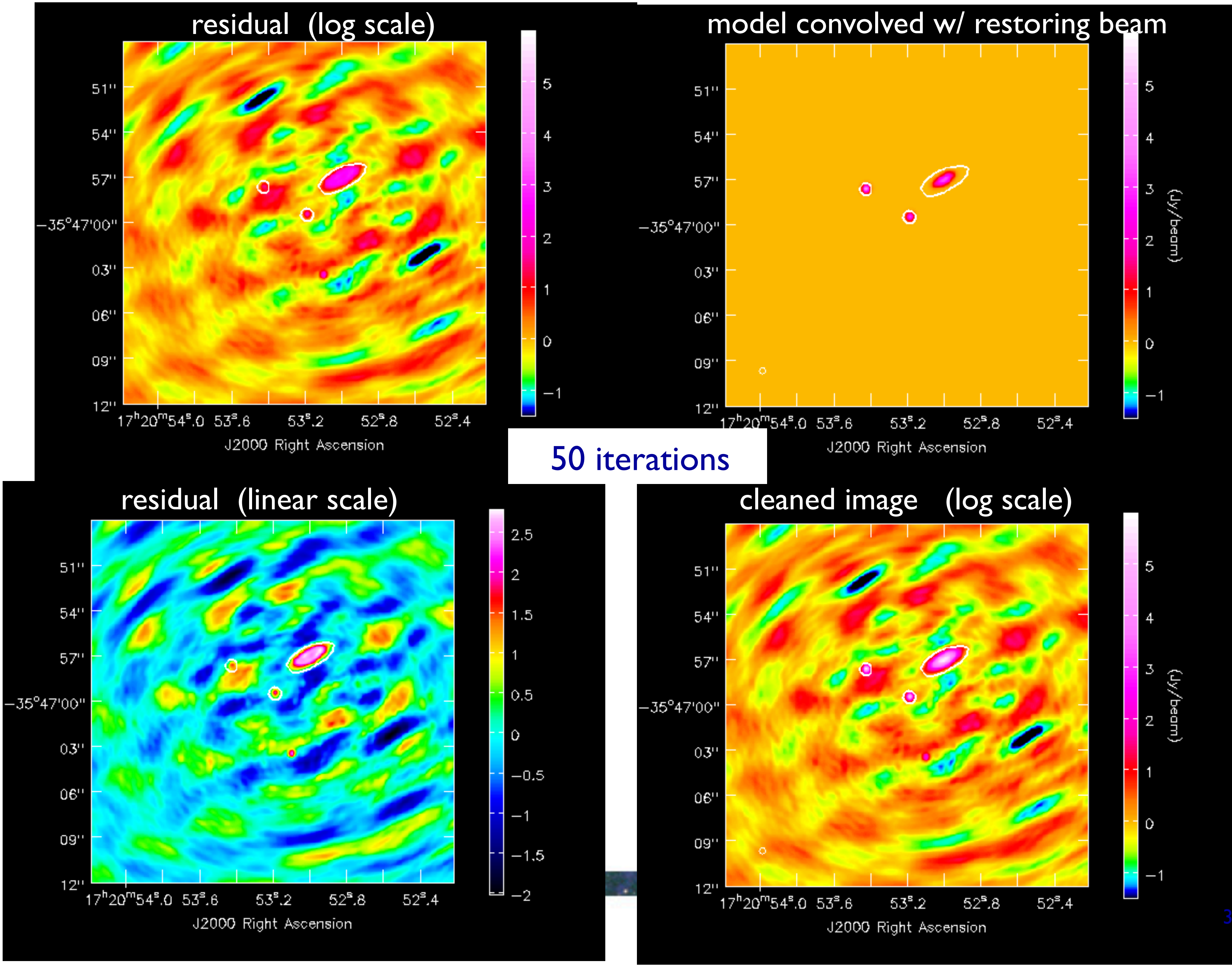


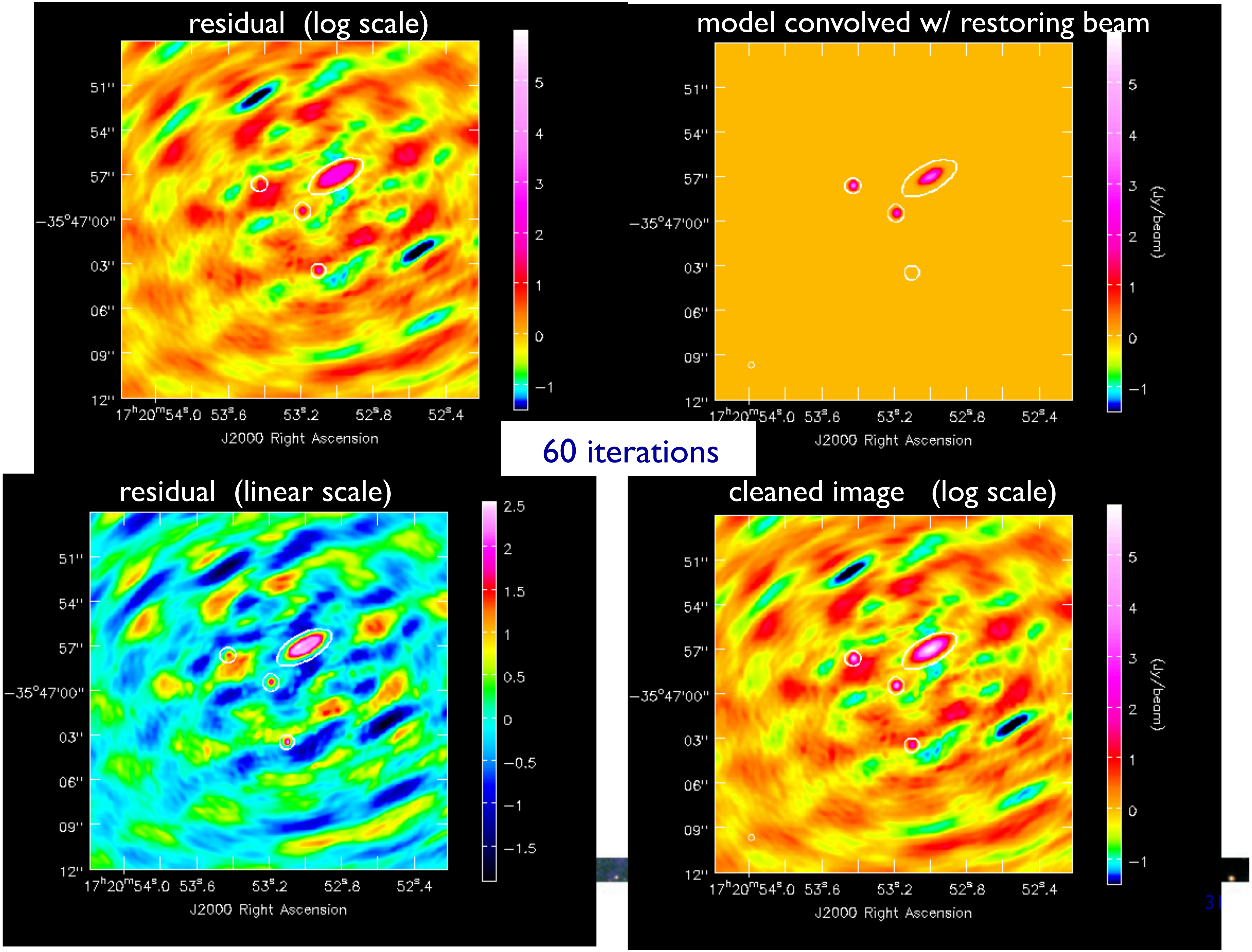


40 iterations

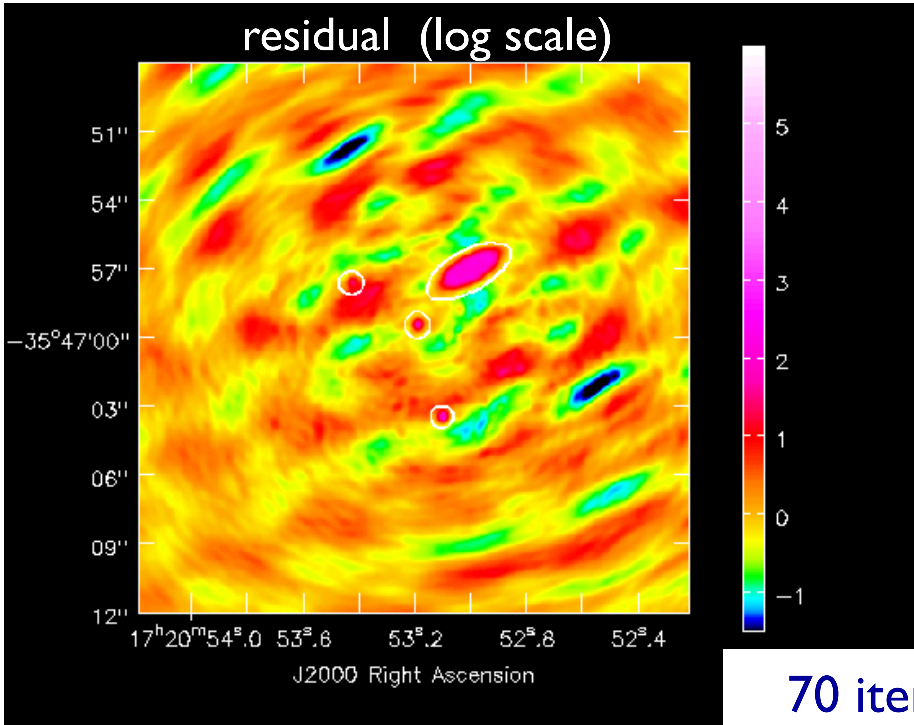


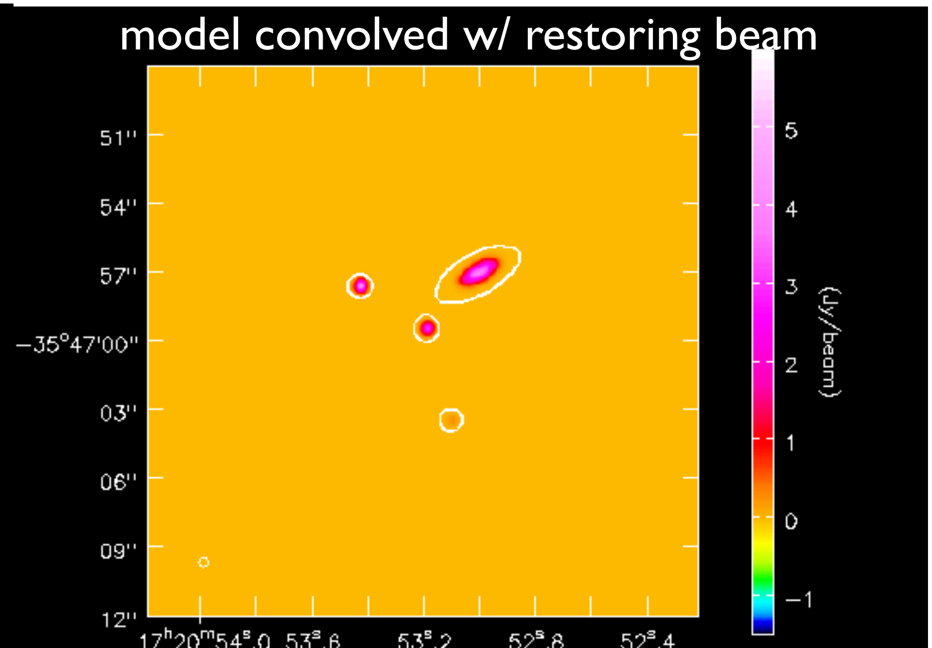



residual (log scale)
model convolved w/ restoring beam
50 iterations
residual (linear scale)
cleaned image (log scale)
J2000 Right Ascension
(Jy/beam)

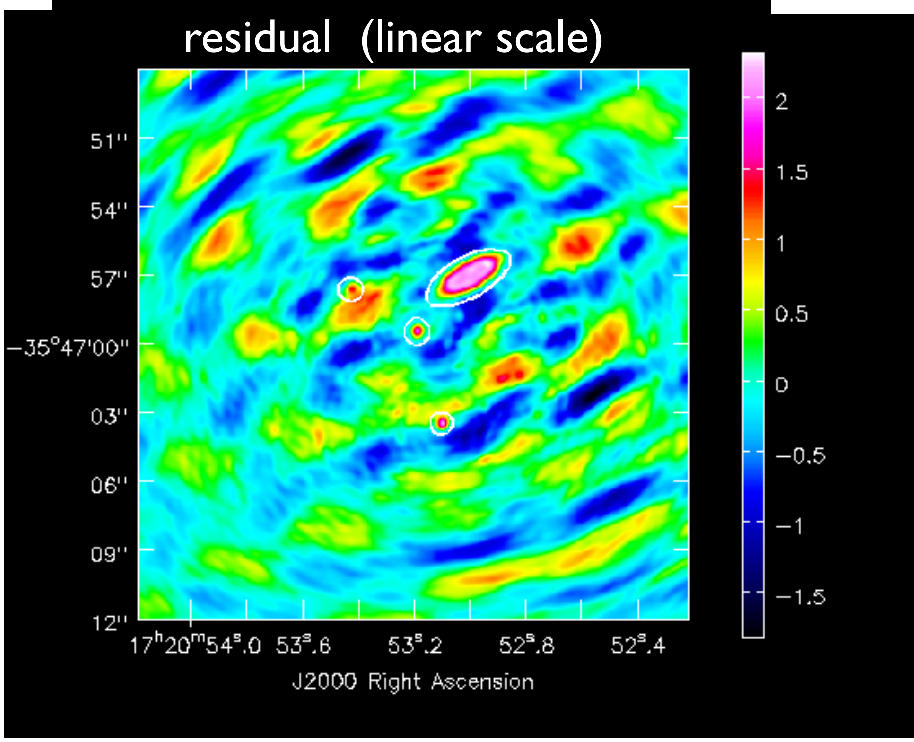


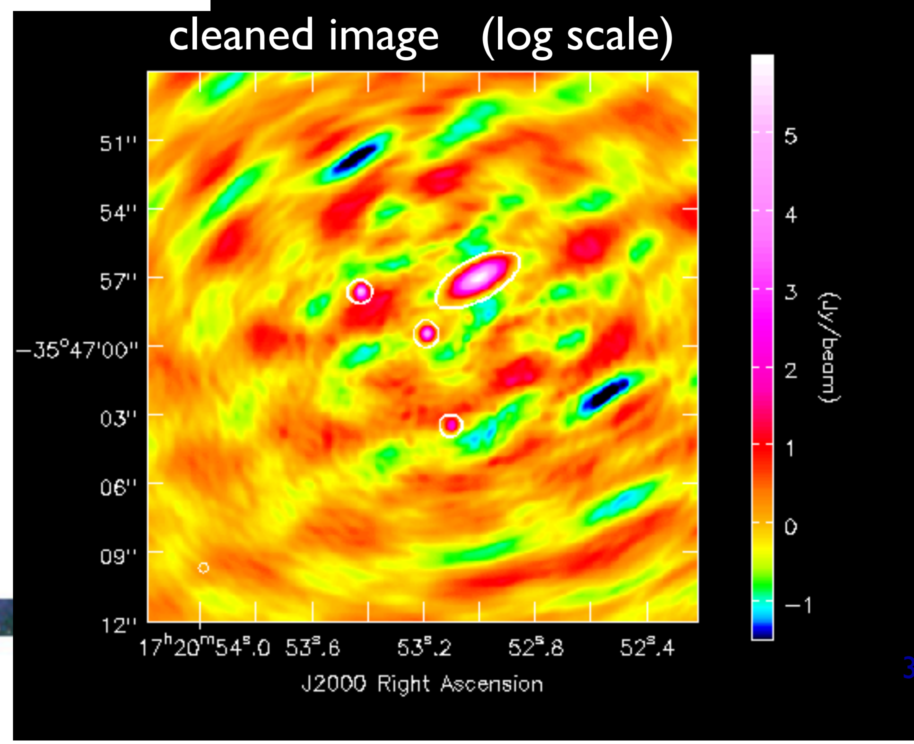
residual (log scale)
model convolved w/ restoring beam
60 iterations
residual (linear scale)
cleaned image (log scale)
J2000 Right Ascension
J2000 Declination
(Jy/beam)
17h20m54s.0 53s.6 53s.2 52s.8 52s.4
51'' 54'' 57'' −35°47'00'' 03'' 06'' 09'' 12''

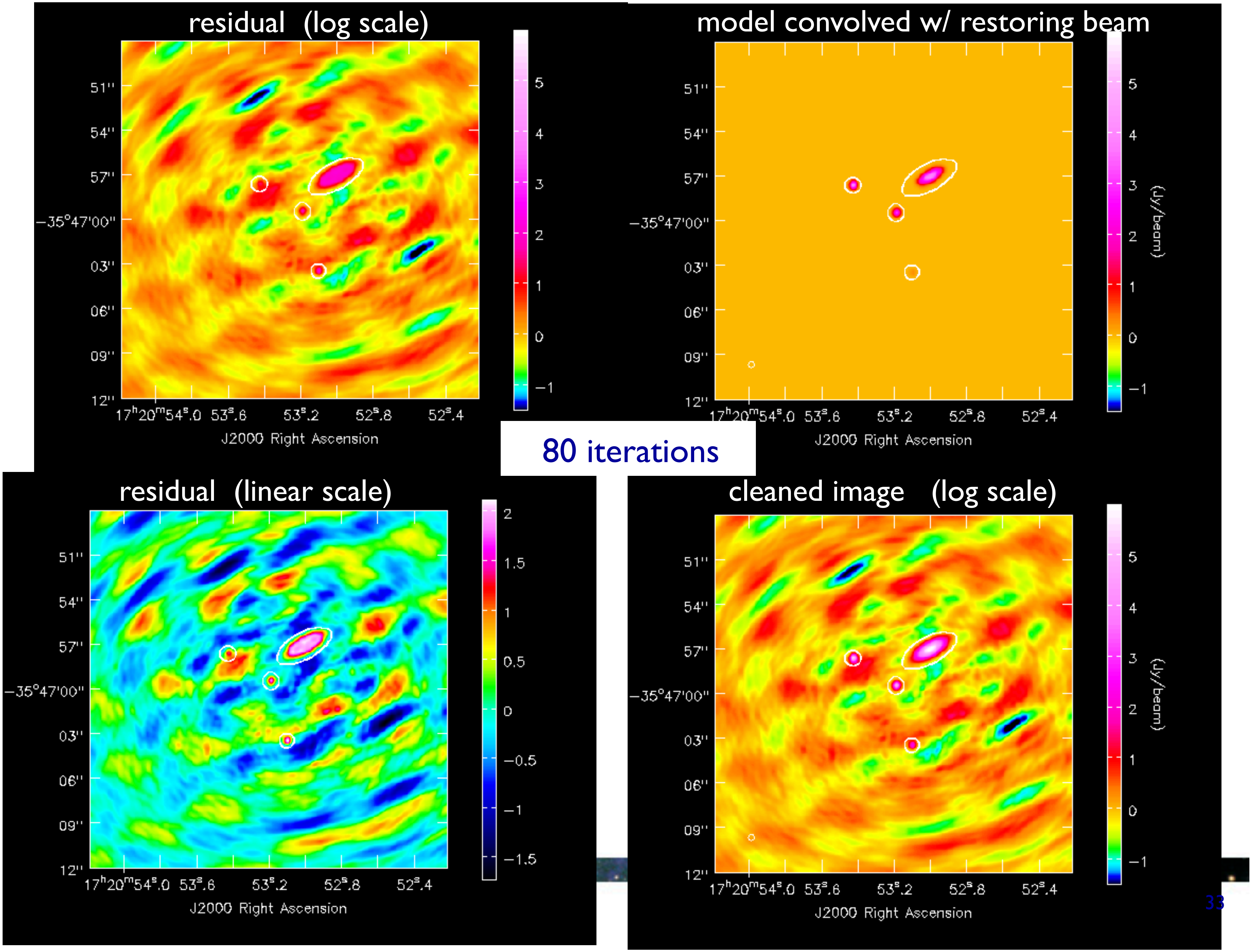


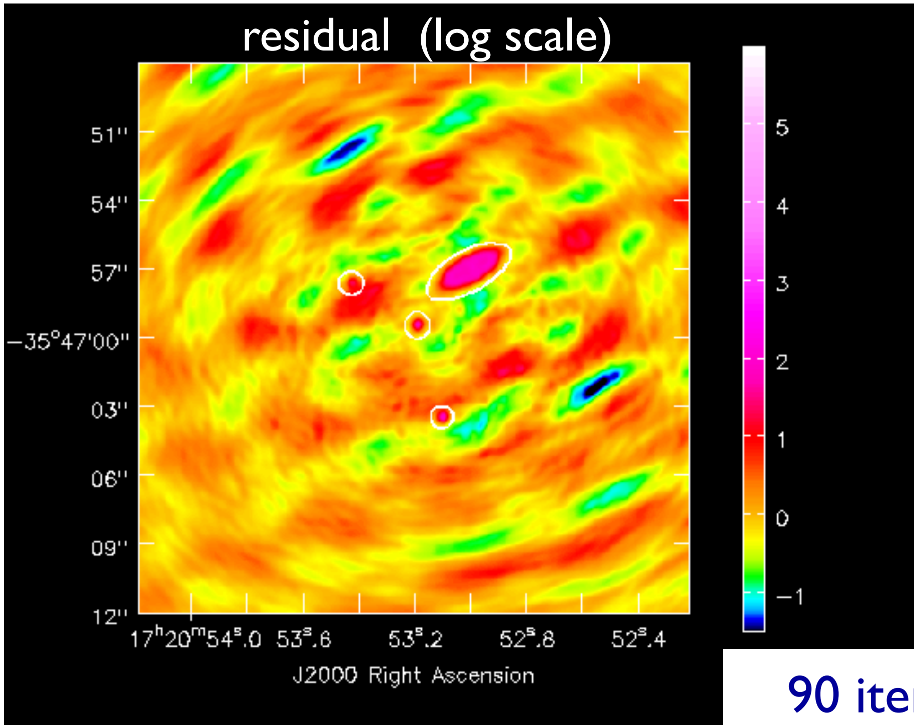


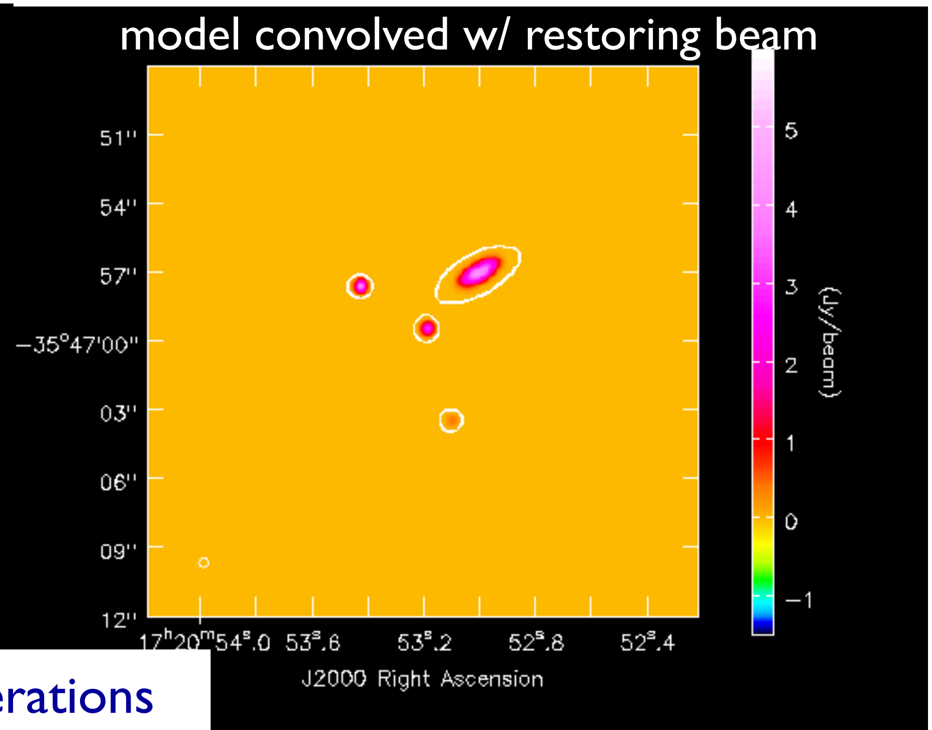
70 iterations



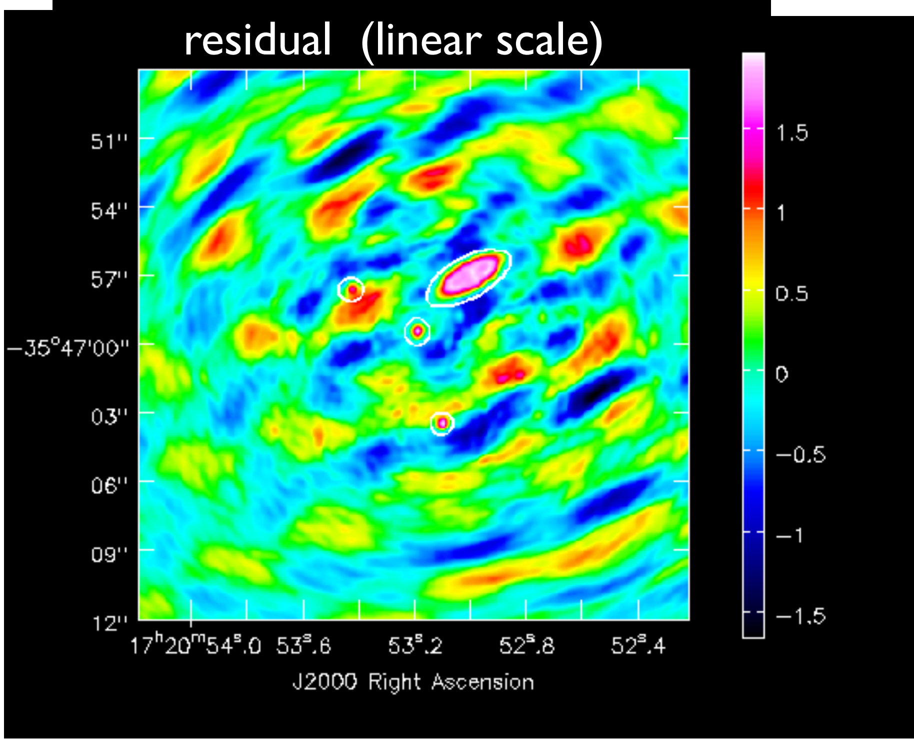


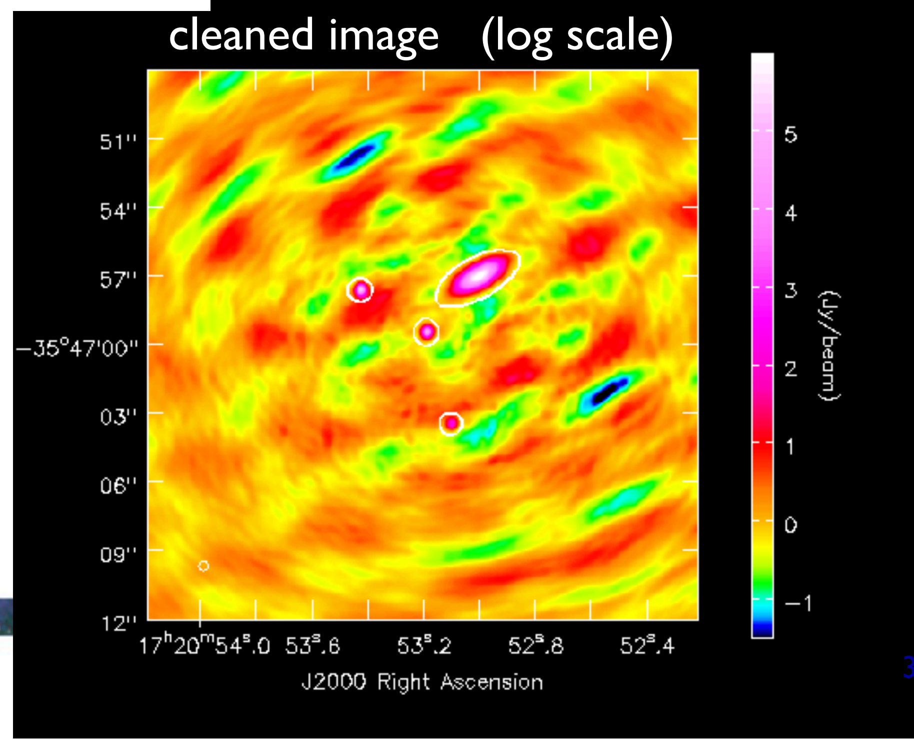

residual (log scale)
model convolved w/ restoring beam
80 iterations
residual (linear scale)
clened image (log scale)
(Jy/beam)
J2000 Right Ascension
−35°47'00"
17h20m54s.0 53s.6 53s.2 52s.8 52s.4
51" 54" 57" 03" 06" 09" 12"

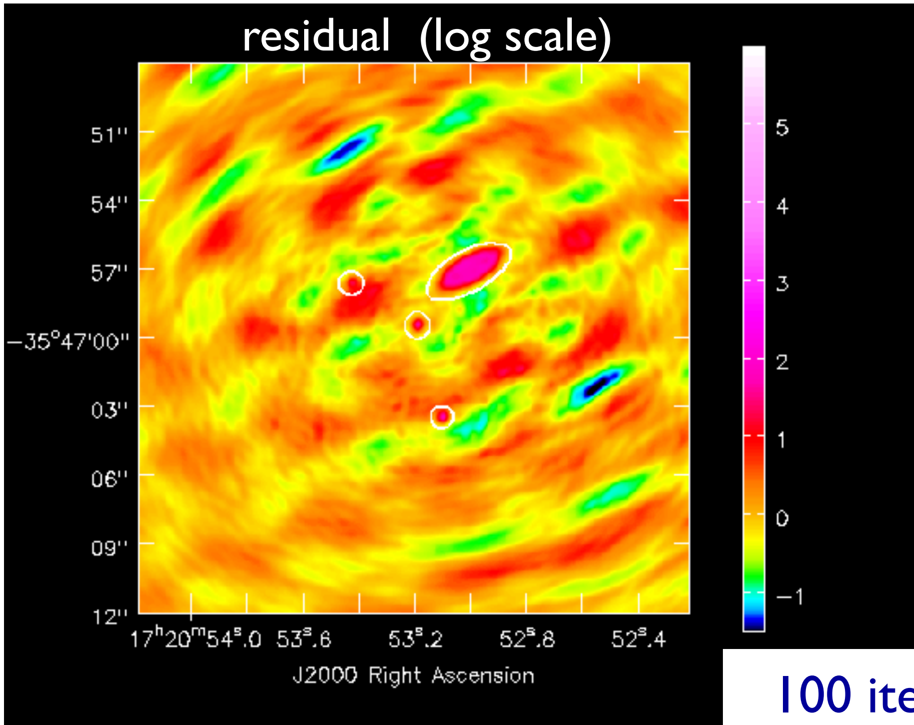


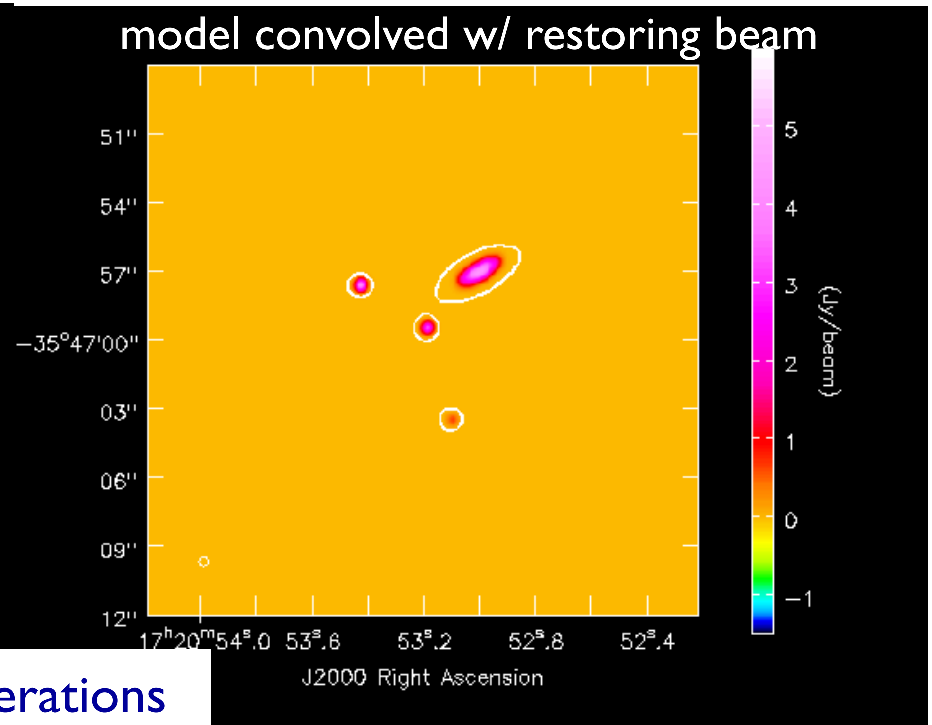


90 iterations

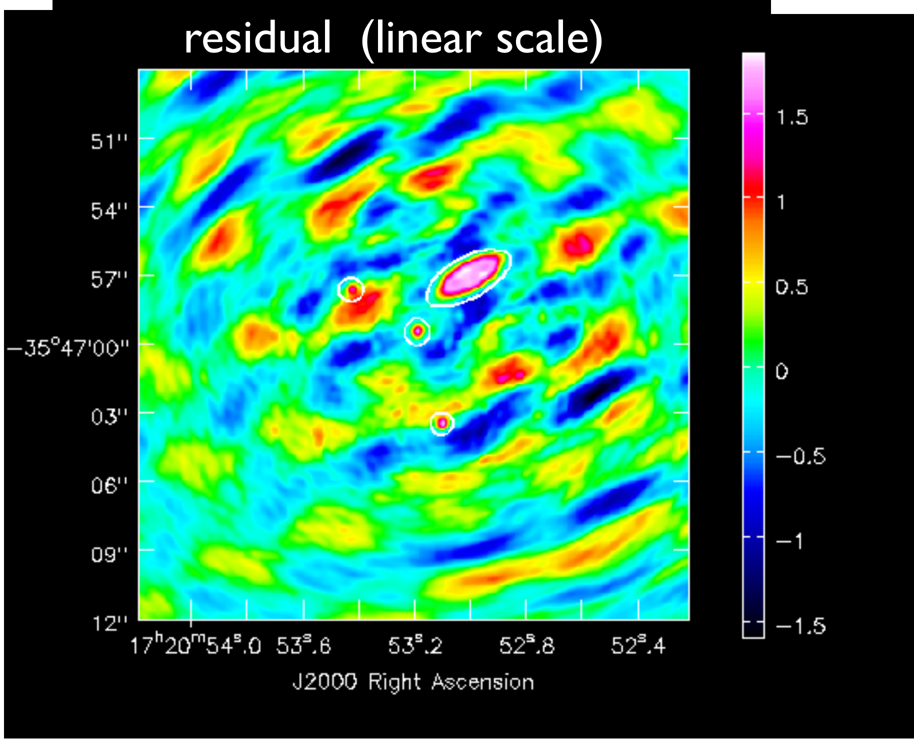


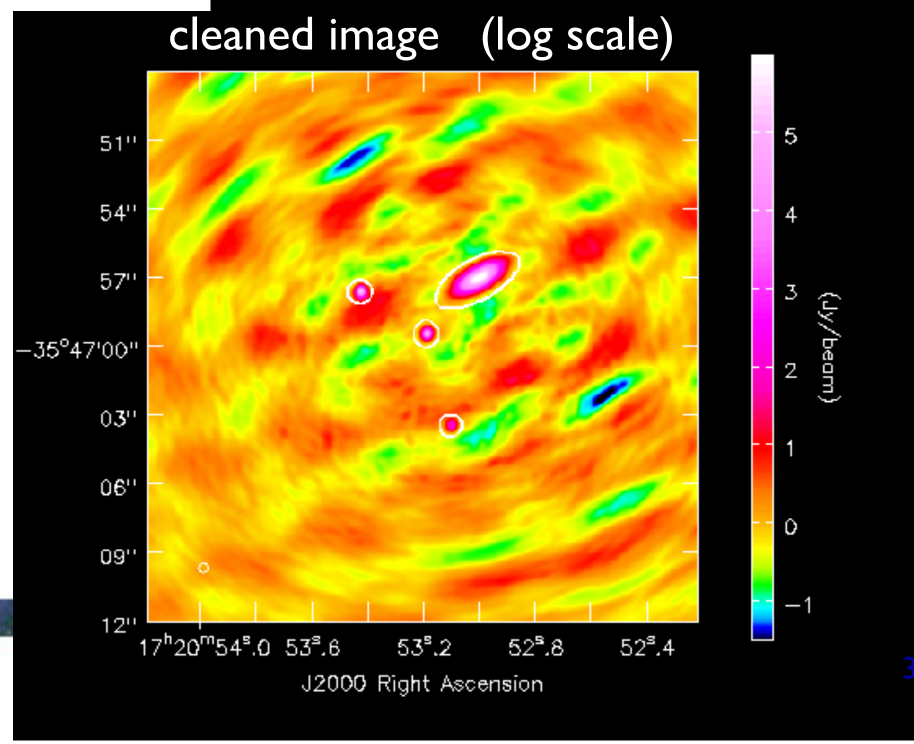


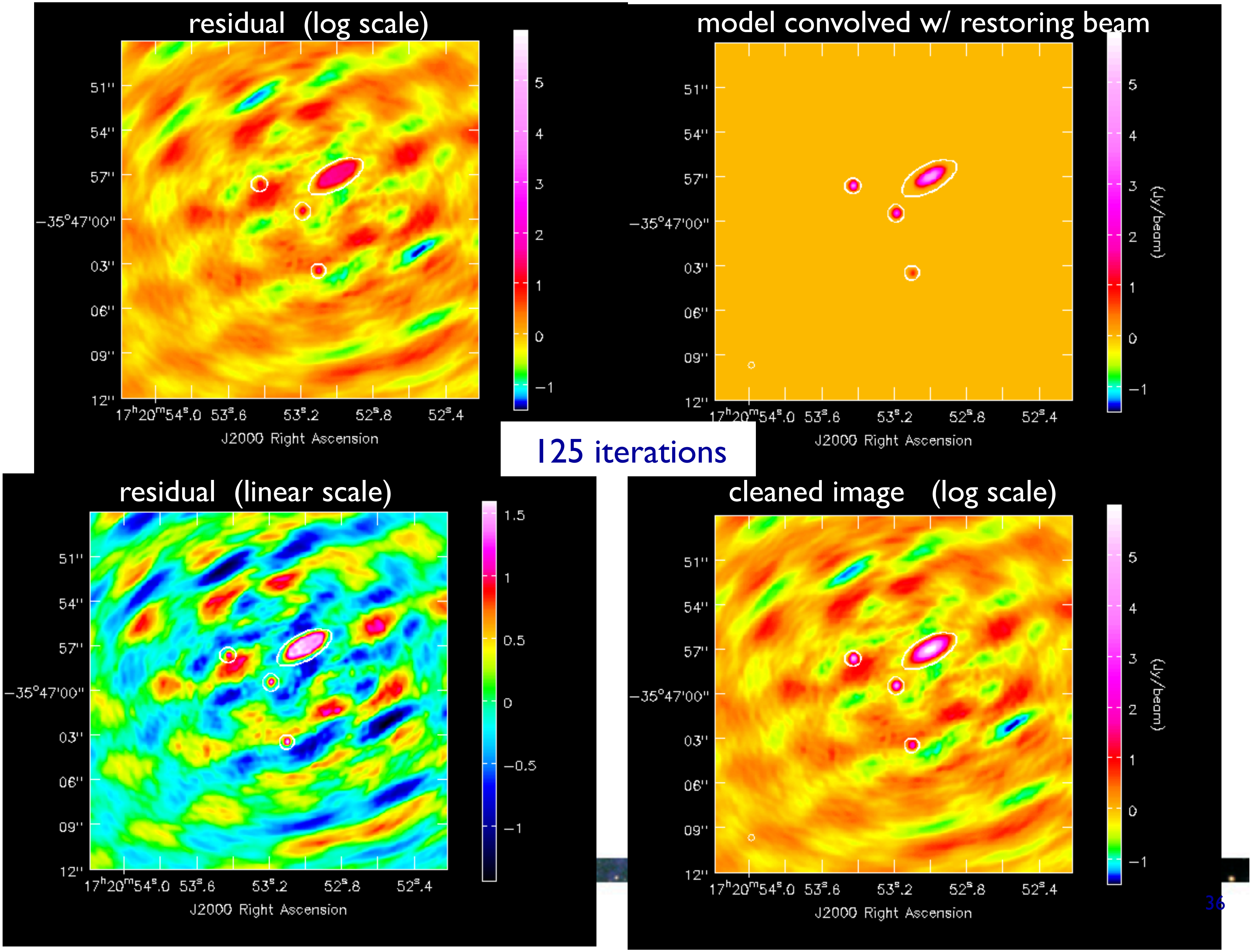




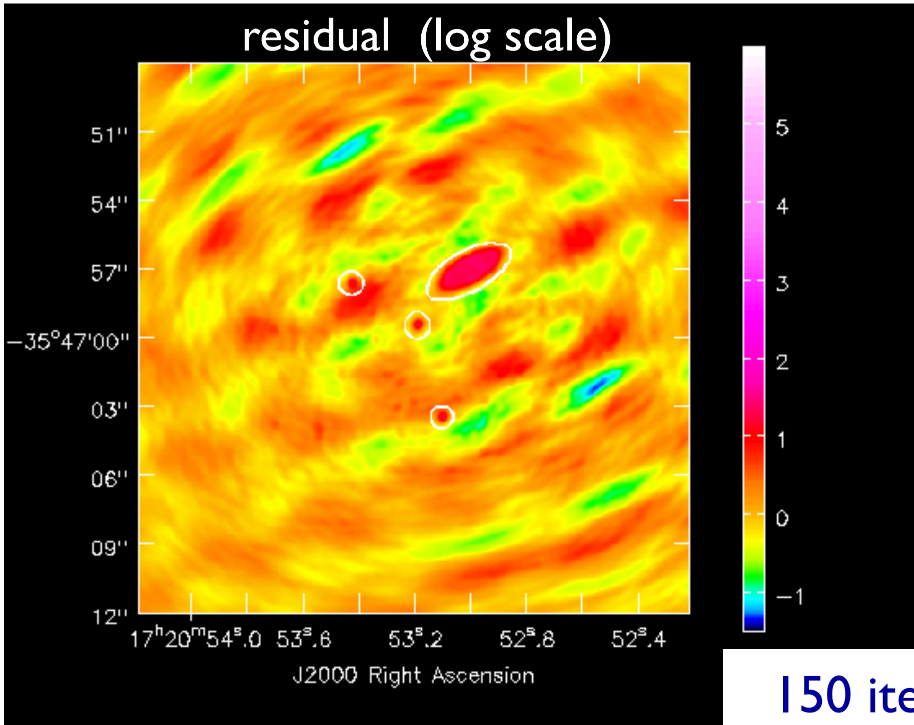
100 iterations

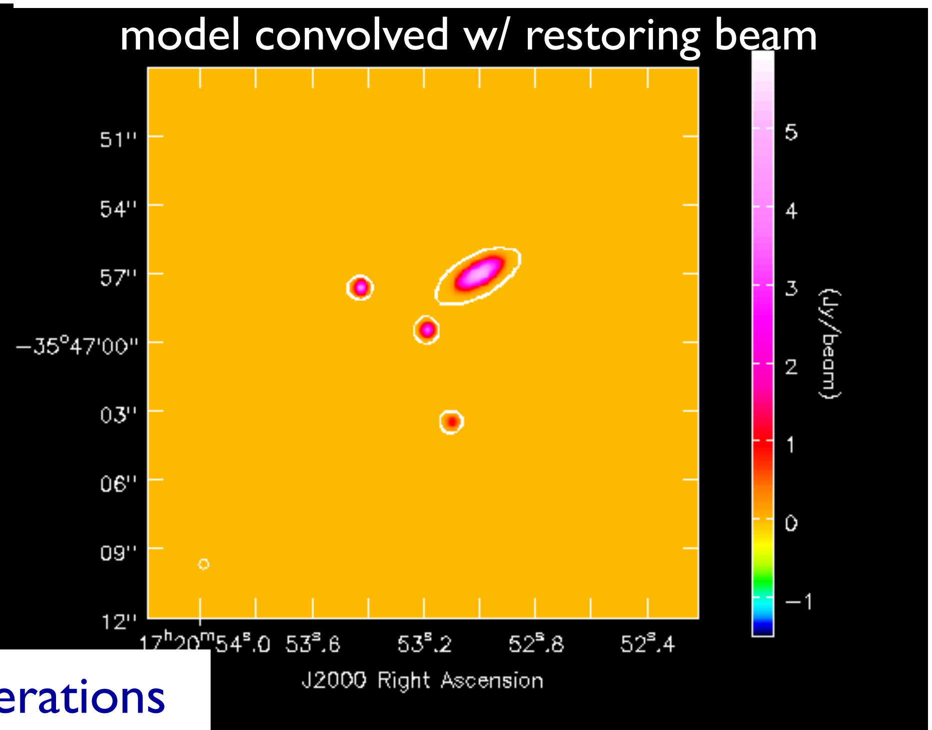




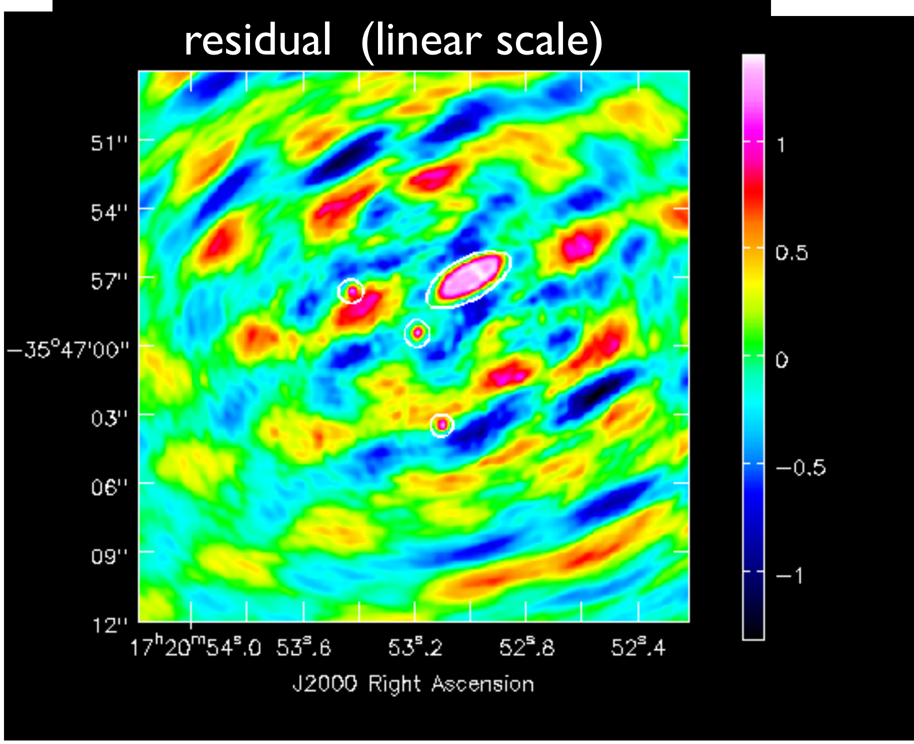


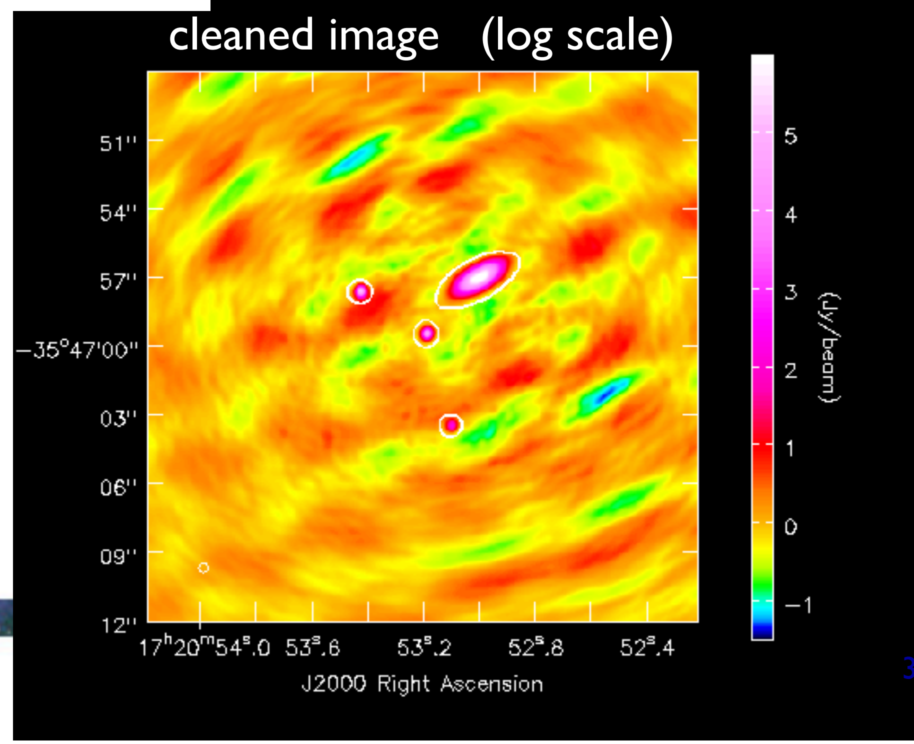
125 iterations



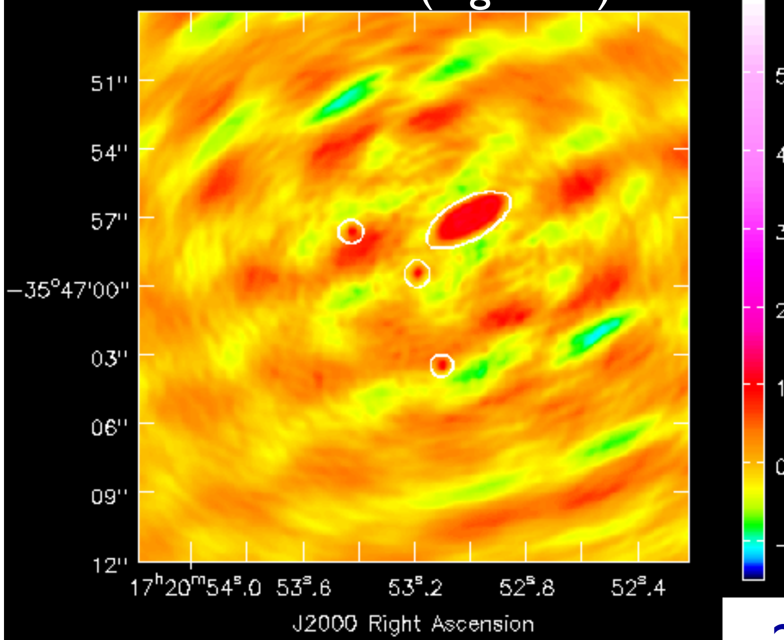


150 iterations

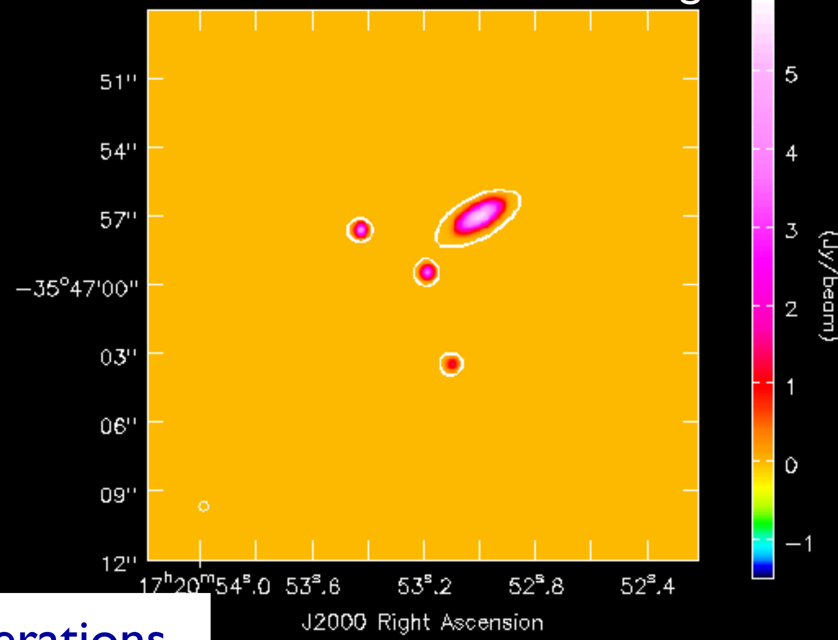




residual (log scale)

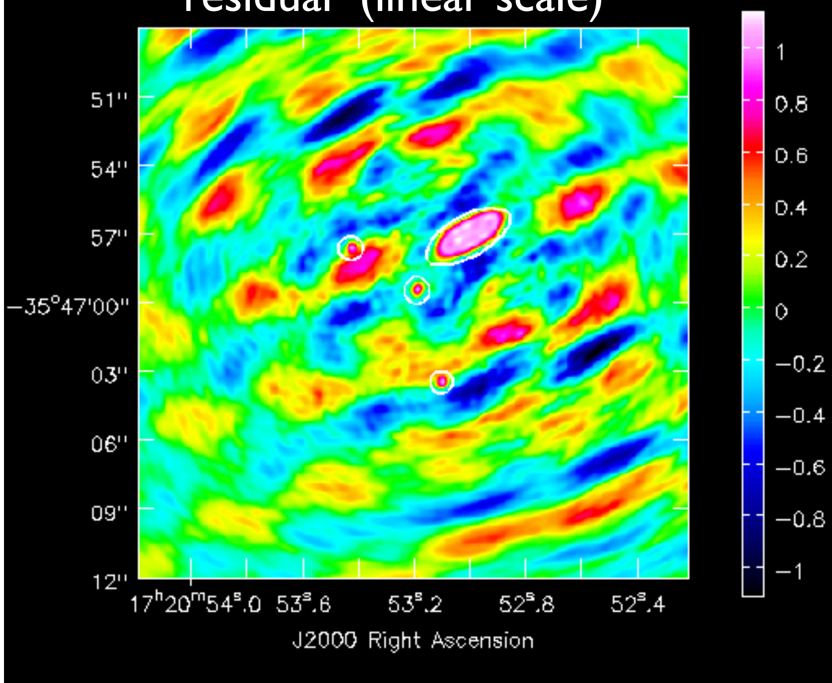


model convolved w/ restoring beam

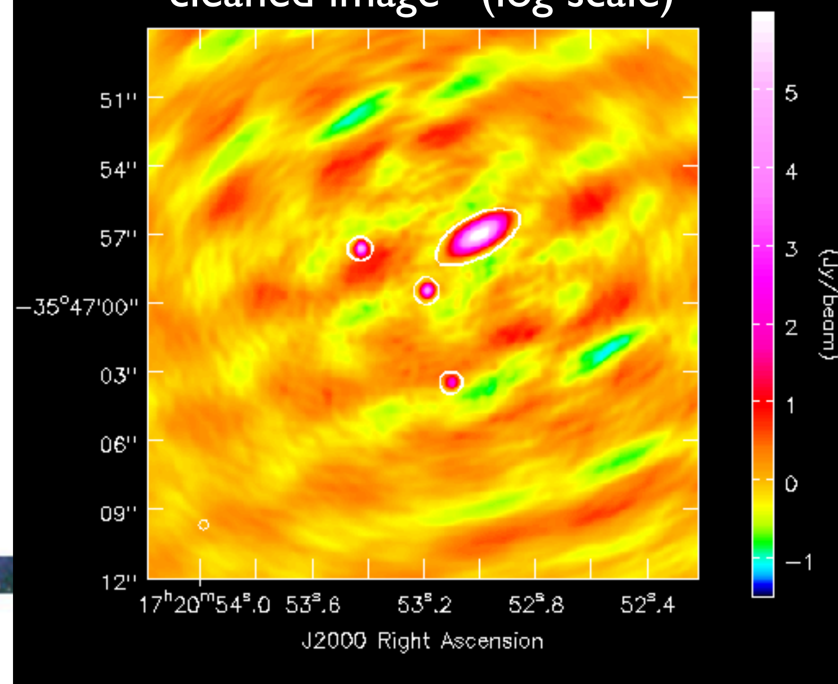


200 iterations

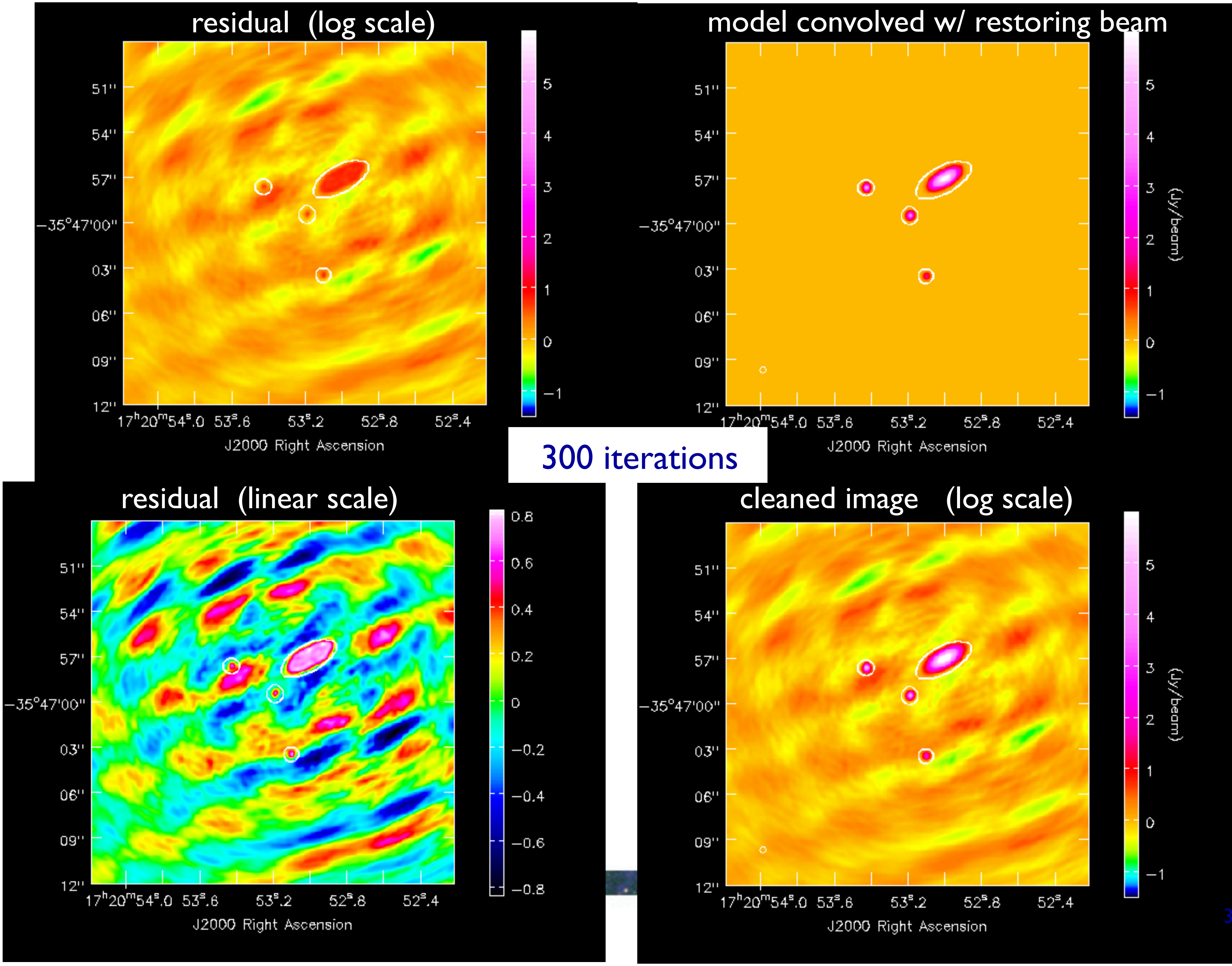
residual (linear scale)

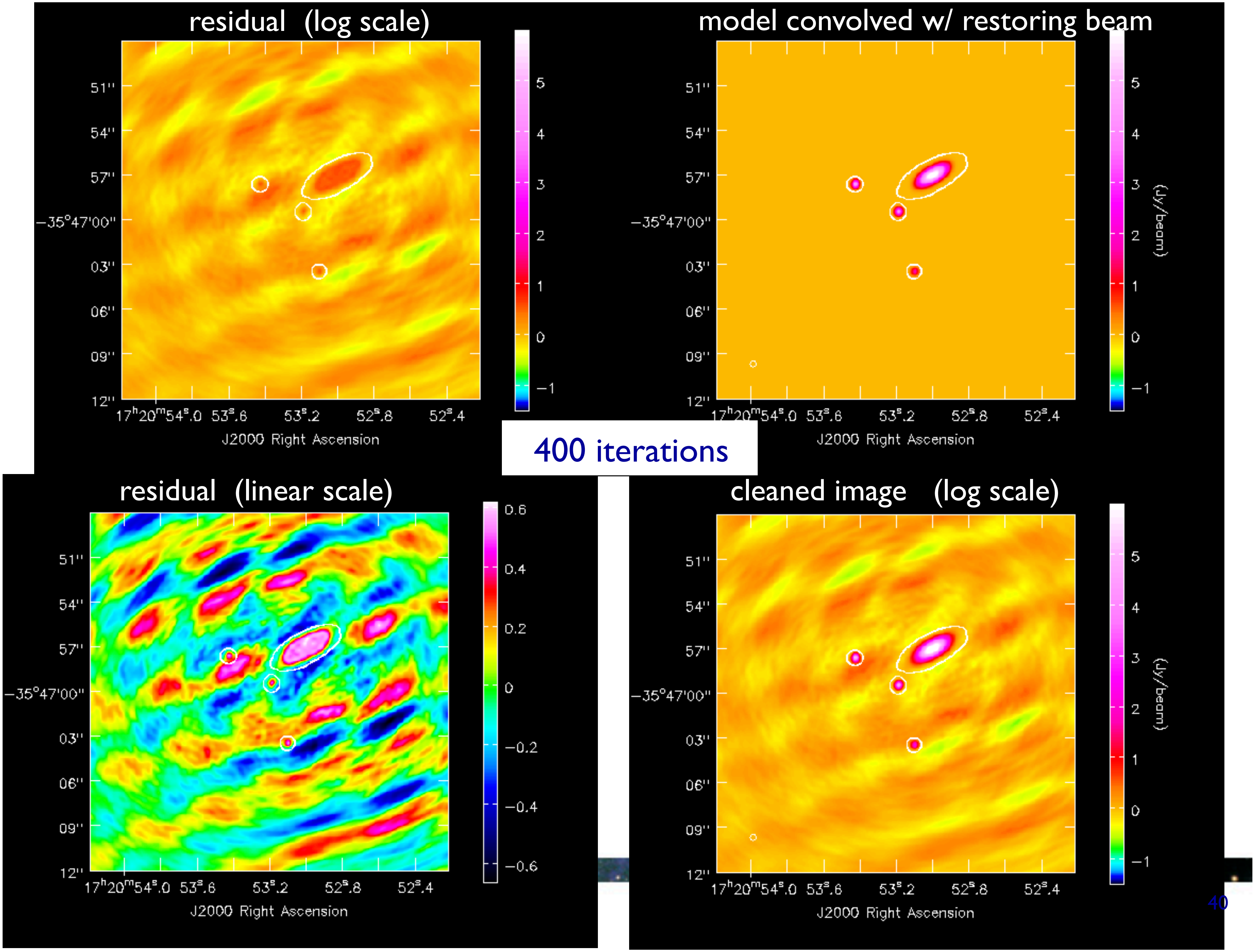


cleaned image (log scale)



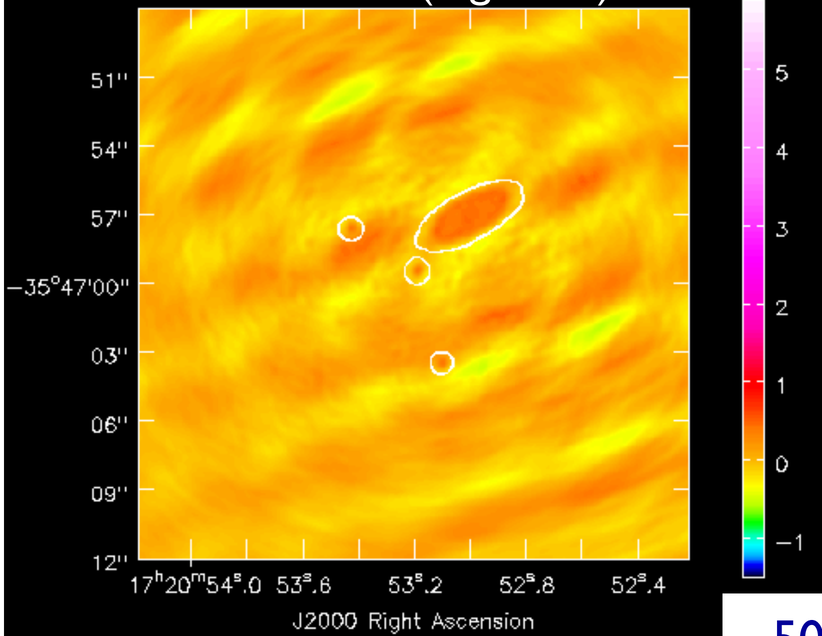
300 iterations



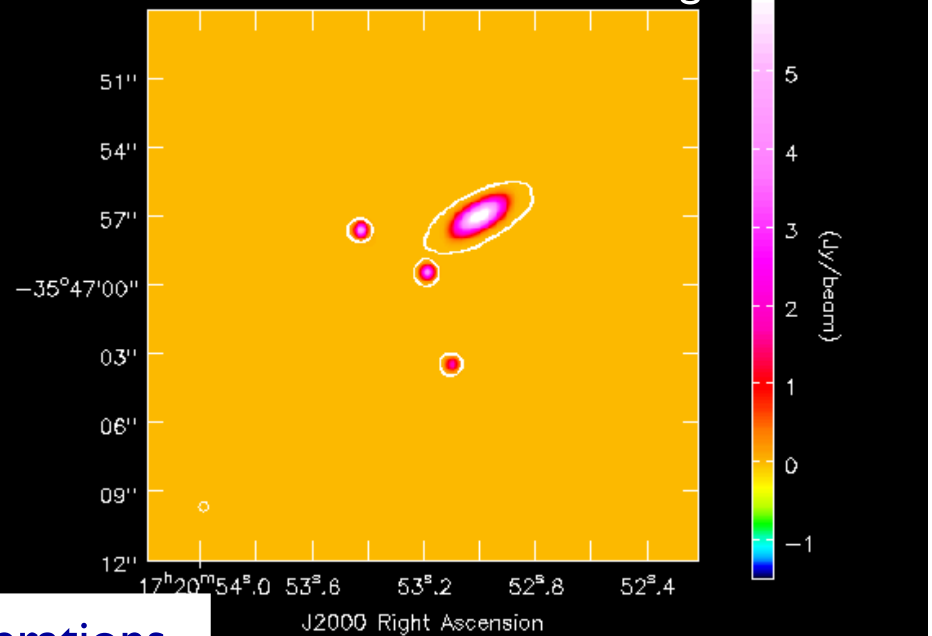


residual (log scale)
model convolved w/ restoring beam
400 iterations
residual (linear scale)
cleaned image (log scale)
J2000 Right Ascension
(Jy/beam)

residual (log scale)

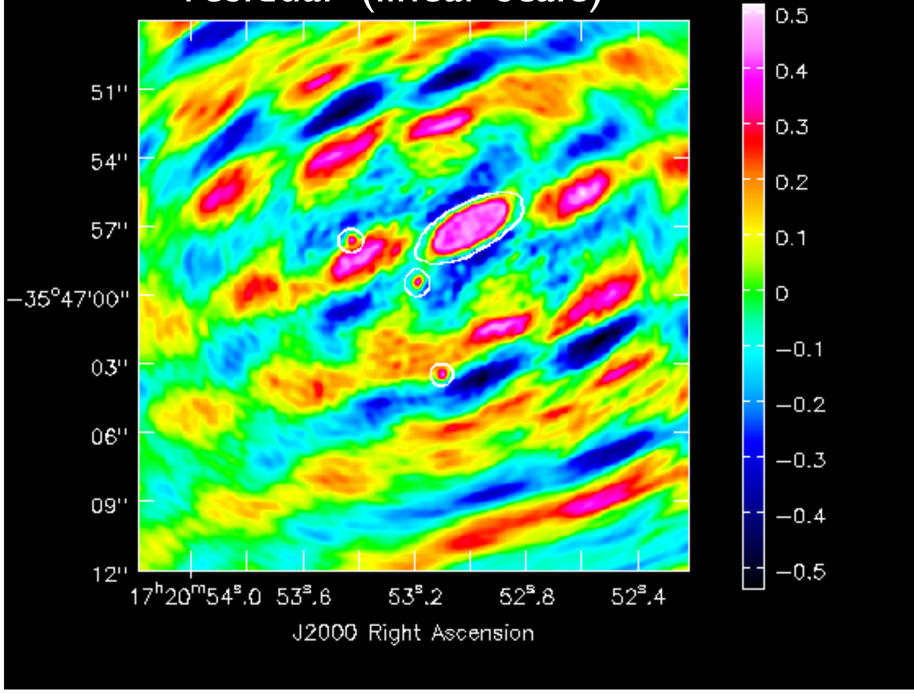


model convolved w/ restoring beam

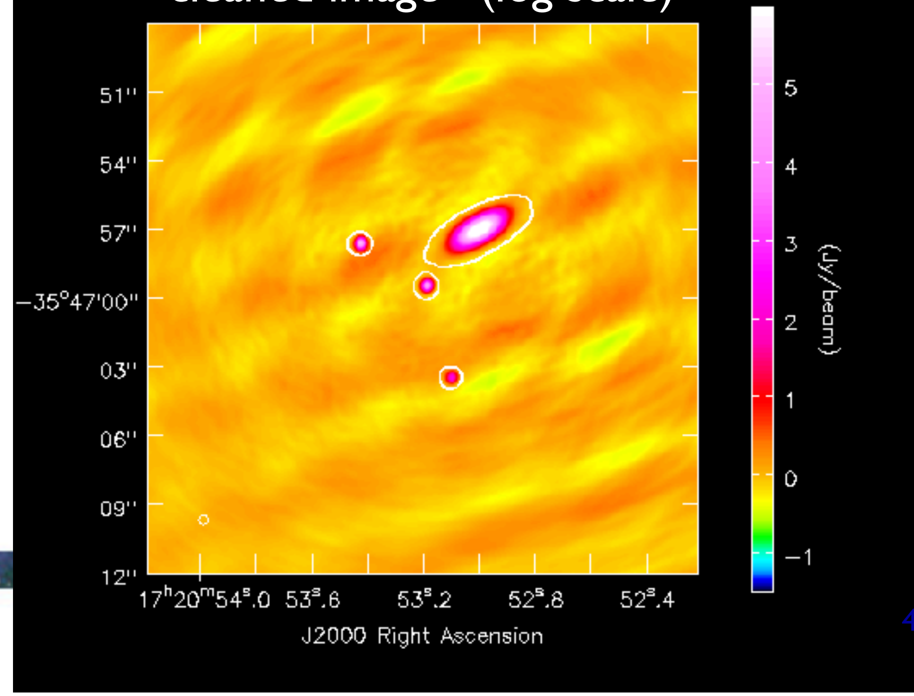


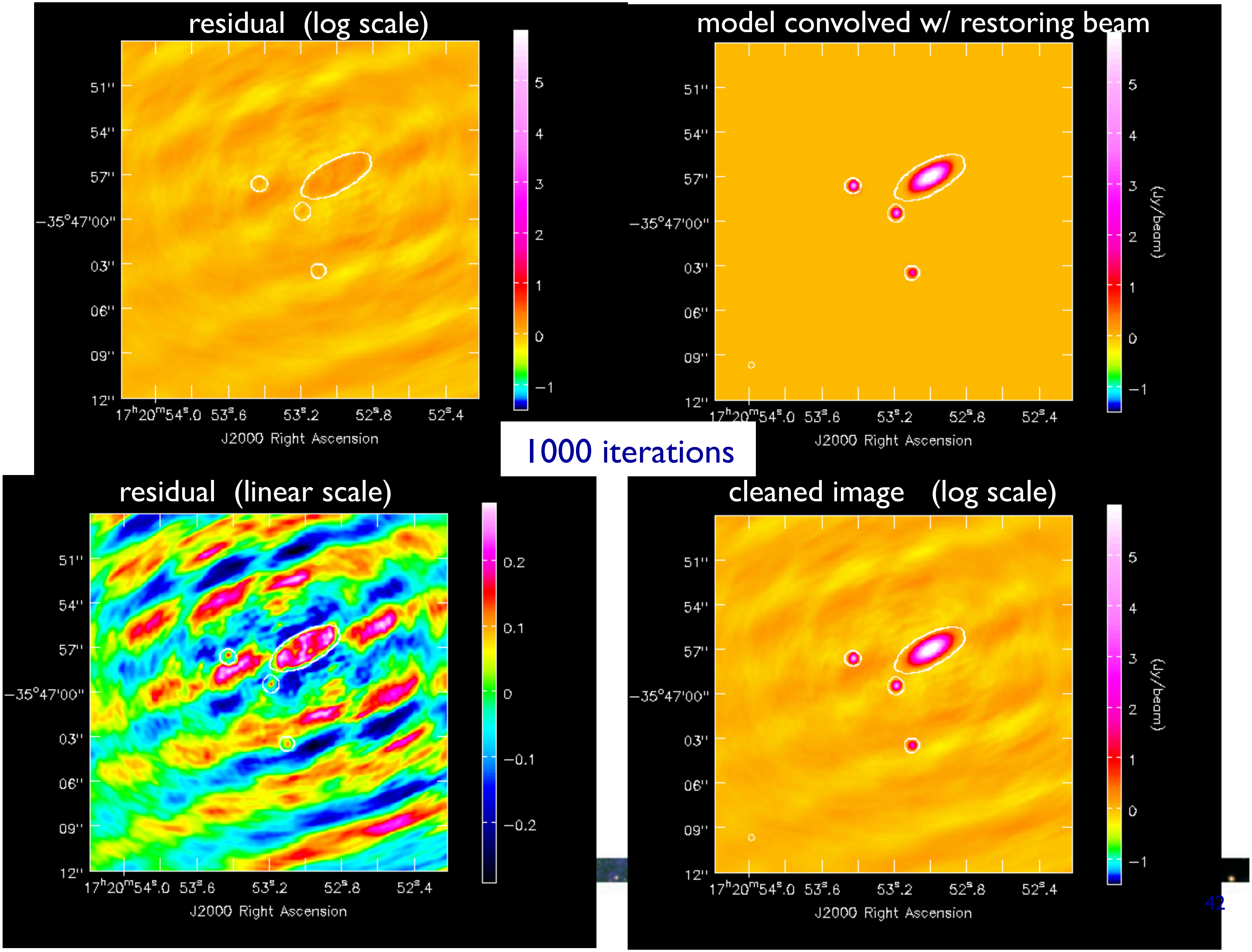
500 iterations

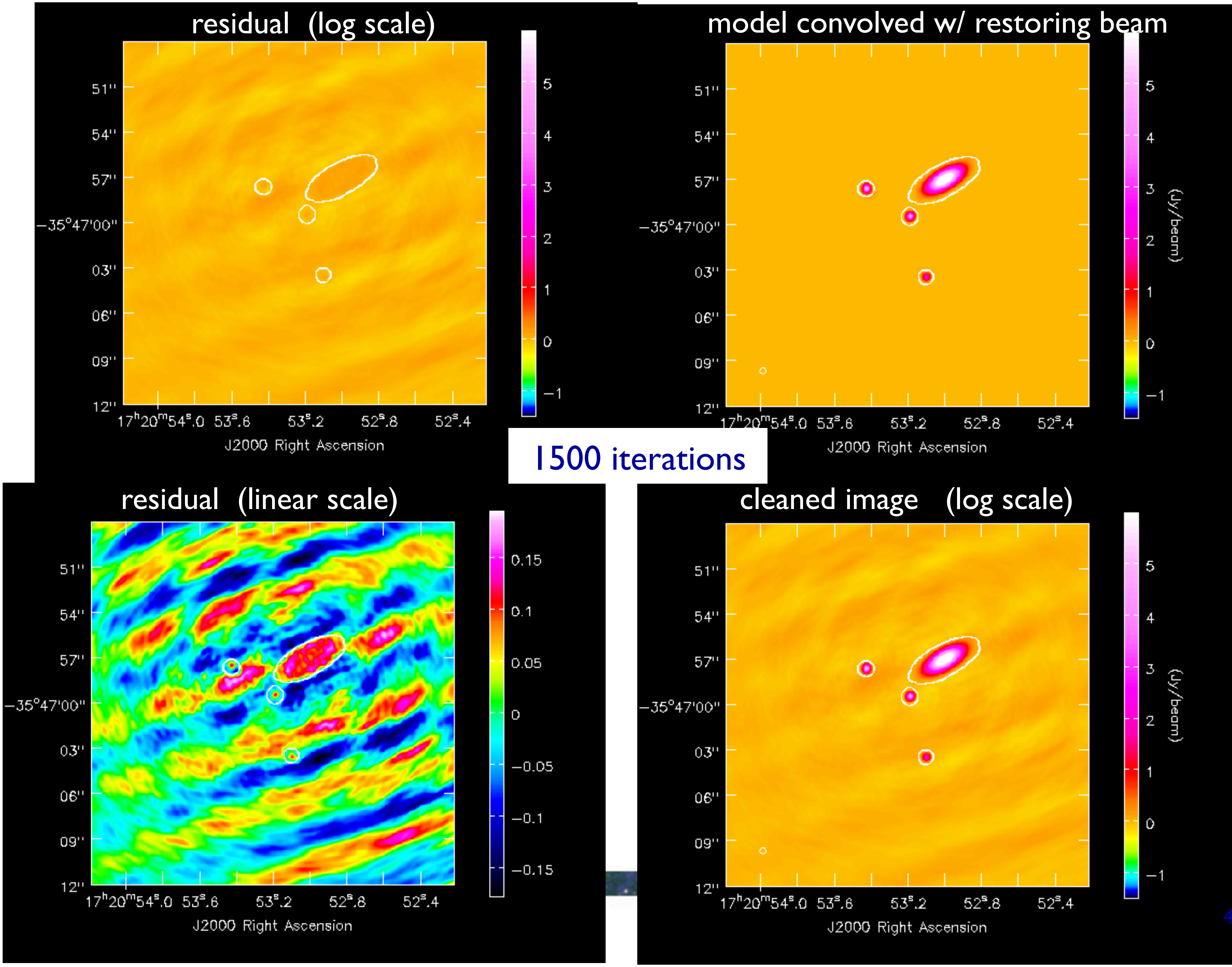
residual (linear scale)



cleaned image (log scale)



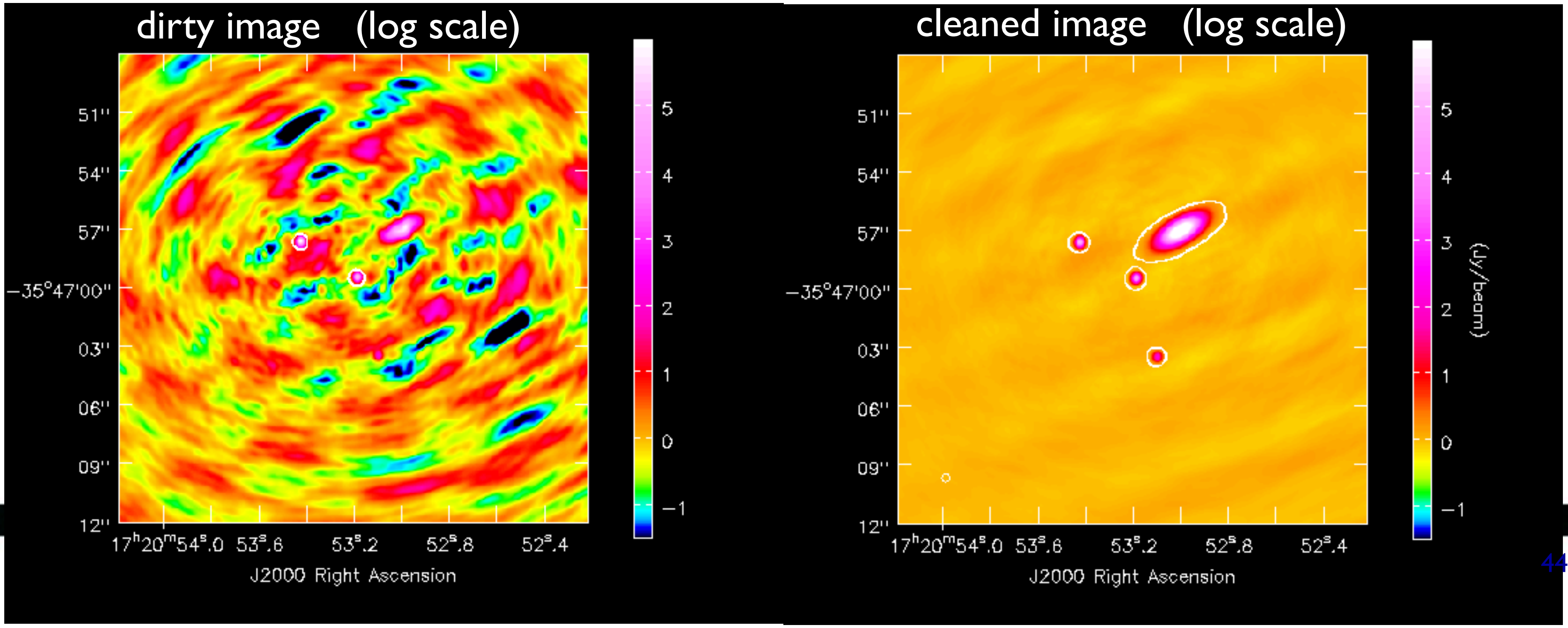

residual (log scale)
model convolved w/ restoring beam
1000 iterations
residual (linear scale)
cleaned image (log scale)
(Jy/beam)
J2000 Right Ascension


residual (log scale)
model convolved w/ restoring beam
1500 iterations
residual (linear scale)
cleaned image (log scale)
J2000 Right Ascension
(Jy/beam)
−35°47'00"
17h20m54s.0 53s.6 53s.2 52s.8 52s.4
51" 54" 57" 03" 06" 09" 12"

Deconvolution

Results depend on:

- image parameters: size, cell, weighting, mosaic, cube
- deconvolution parameters: algorithm, iterations, boxing, stopping criteria



Deconvolution

- to keep you awake at night
  - $\exists$ an infinite number of T(x,y) compatible with sampled V(u,v), i.e. "invisible" distributions R(x,y) where $b(x,y) \otimes R(x,y) = 0$
    - no data beyond $u_{max}, v_{max} \rightarrow$ unresolved structure
    - no data within $u_{min}, v_{min} \rightarrow$ limit on largest size scale
    - holes between u_{min}, v_{min} and $u_{max}, v_{max} \rightarrow$ sidelobes
  - noise $\rightarrow$ undetected/corrupted structure in T(x,y)
  - no unique prescription for extracting optimum estimate of true sky brightness from visibility data
- deconvolution
  - uses non-linear techniques effectively interpolate/extrapolate samples of V(u,v) into unsampled regions of the (u,v) plane
  - aims to find a **sensible** model of T(x,y) compatible with data
  - requires *a priori* assumptions about T(x,y)

NRAO

the clean task in casapy:

```
--------> 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          =         ''        #  Options: 'csclean' or 'mosaic', '', uses psfmode
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
calready            =       True        #  True required for self-calibration
allowchunk          =      False        #  Divide large image cubes into channel chunks for deconvolution
async               =      False        #  If true the taskname must be started using clean(...)
```

the clean task in casapy:

```
--------> 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          =        ''        #  Options: 'csclean' or 'mosaic', '', uses psfmode
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
calready            =      True        #  True required for self-calibration
allowchunk          =     False        #  Divide large image cubes into channel chunks for deconvolution
async               =     False        #  If true the taskname must be started using clean(...)
```

- (visibility) data selection
- gridding & inversion
- deconvolution
- basic image parameters

visibility data selection

```
#  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
```

- searches by name before index
- field >1 MUST use imagermode=mosaic

Basic Image Parms: Pixel Size and Image Size

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

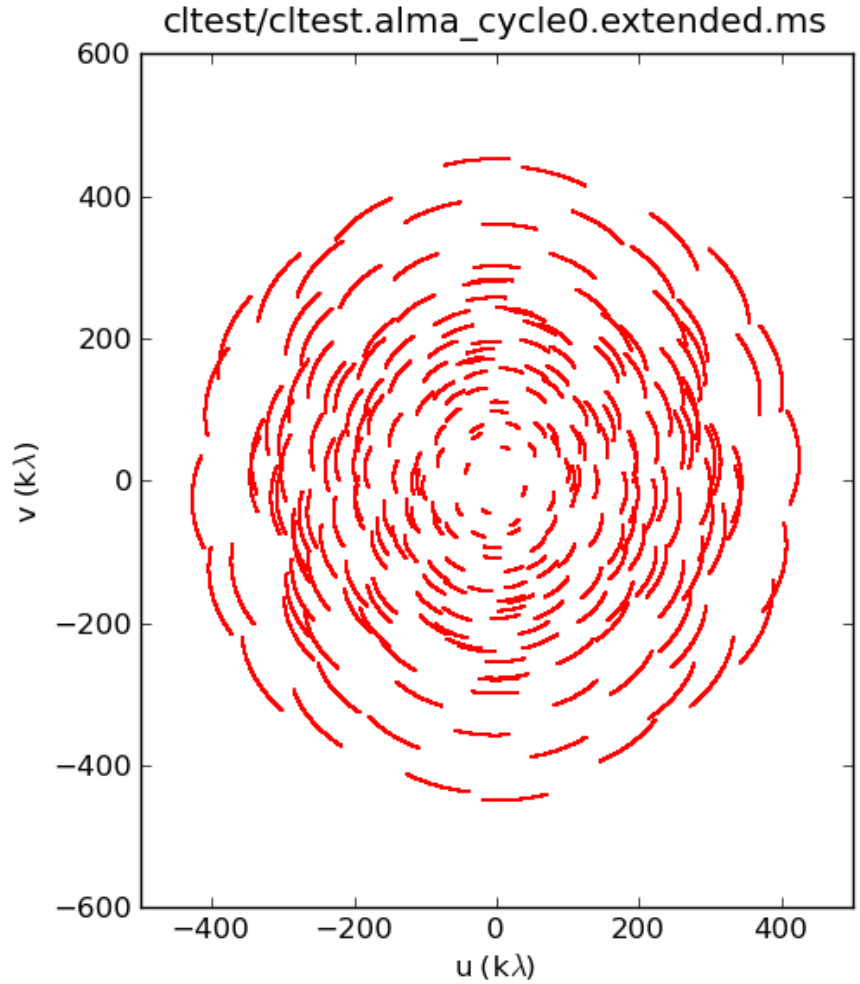
- image size
  - natural resolution in (u,v) plane samples FT{A(x,y)}, implies image size 2x primary beam
  - e.g., ALMA: 870 μm, 12 m telescope → 2x 18 arcsec

* not restricted to powers of 2 (in fact internal padding complicates)

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

plotuv

```
#  plotuv :: Plot the baseline distribution
vis                 =         ''         #  Name of input visibili
field               =         ''         #  Select field using ID(
antenna             =         ''         #  Select data based on a
spw                 =         ''         #  Select spectral window
observation         =         ''         #  Select by observation
array               =         ''         #  Select (sub)array(s) b
maxnpts             =     100000         #  Maximum number of poin
colors              = ['r', 'y', 'g', 'b'] #  a list of matplotli
symb                =        ','         #  A matplotlib plot symb
ncycles             =          1         #  How many times to cycl
                                         #   plot.
figfile             =         ''         #  Save the plotted figur
```



im.advise

```
CASA <83>: im.open("ghii_b3/ghii_b3.alma_cycle0.compact.ms")
im  Out[83]: True

CASA <84>: im.advise(takeadvice=False)
  Out[84]:
{'cell': {'unit': 'arcsec', 'value': 2.813915604325822},
 'facets': 1L,
 'phasecenter': 'J2000 05:39:19.21 -069.03.58.345',
 'pixels': 64L,
 'return': True}

CASA <85>: im.done()
  Out[85]: True
```

```
imager::advise()        Advising image properties
imager::advise()        Maximum uv distance = 36650.9 wavelengths
imager::advise() +      Recommended cell size < 2.81392 arcsec
imager::advise()        Recommended number of pixels = 64
imager::advise() +      Dispersion in uv, w distance = 16903.4, 9649.15 wavelengths
imager::advise() +      Best fitting plane is w = -0.00245685 * u + -1.0421 * v
imager::advise() +      Dispersion in fitted w = 605.854 wavelengths
imager::advise() +      Wide field cleaning is not necessary
```

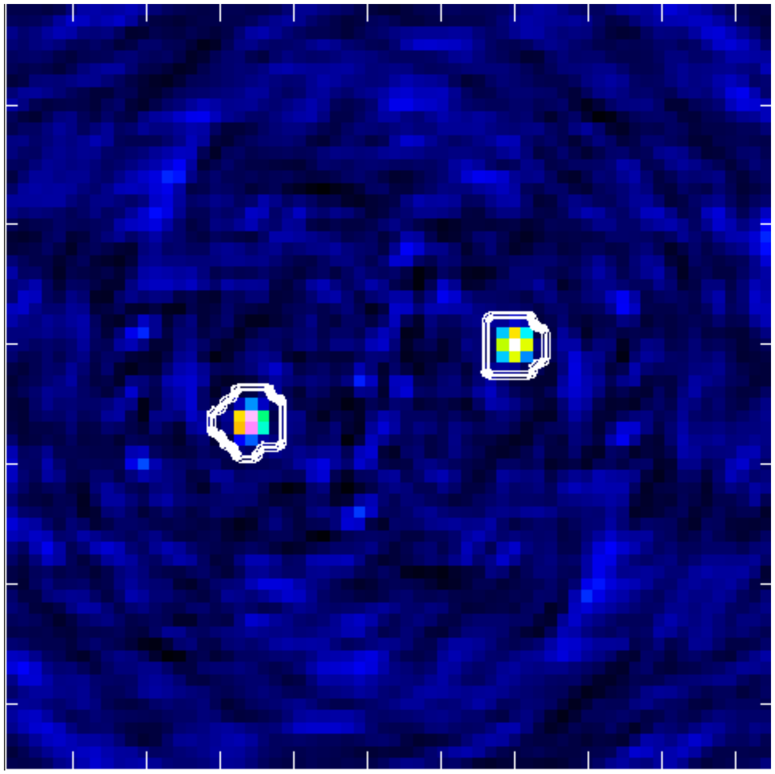
0.5/maxuv – absolute minimum;

recommend < 0.2 = 1.1arcsec

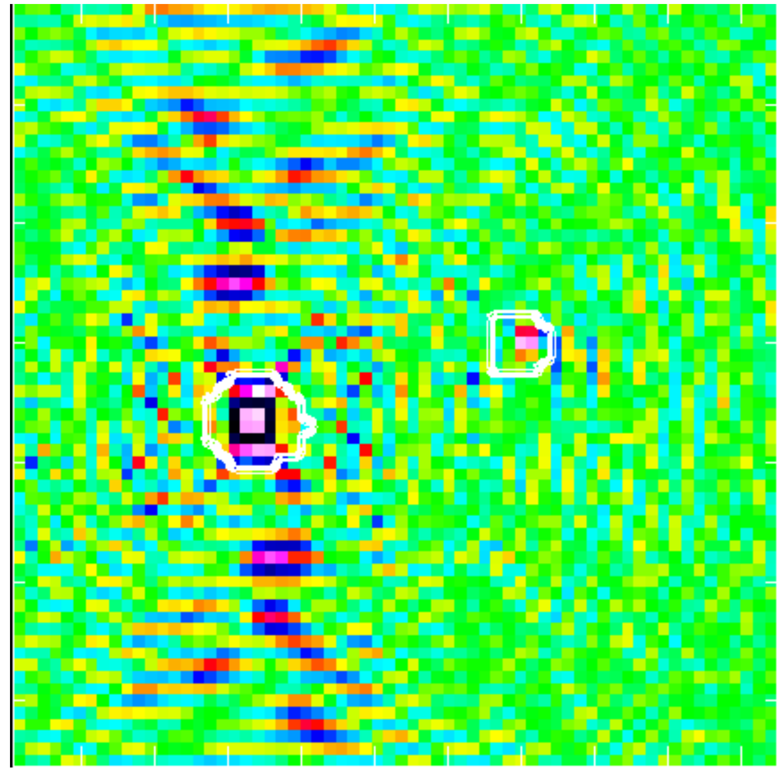
Why sample the PSF so finely?

- Gridding prevents arbitrary placement of model components

dirty image of 2 pt sources



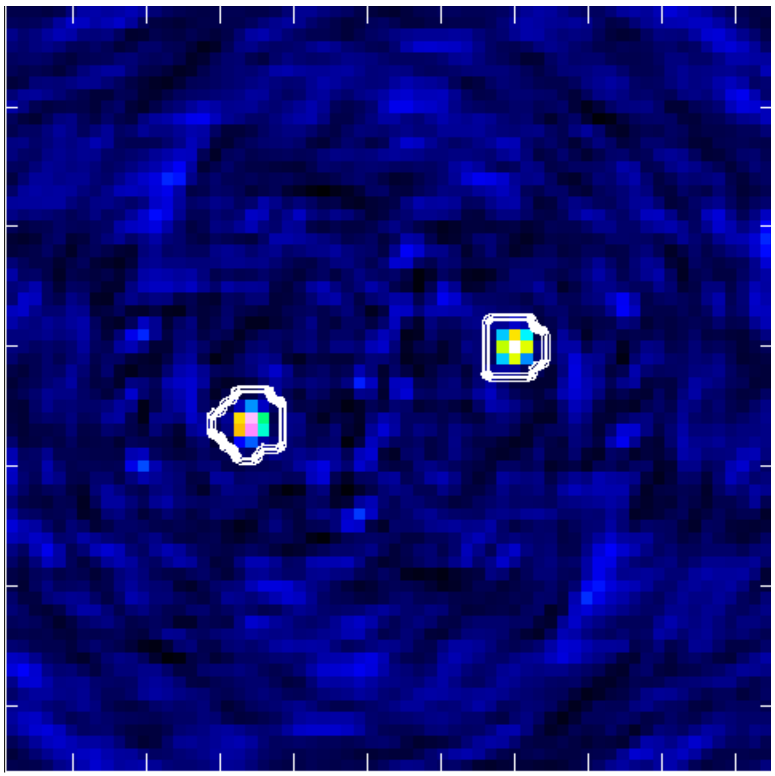
residual after cleaning both

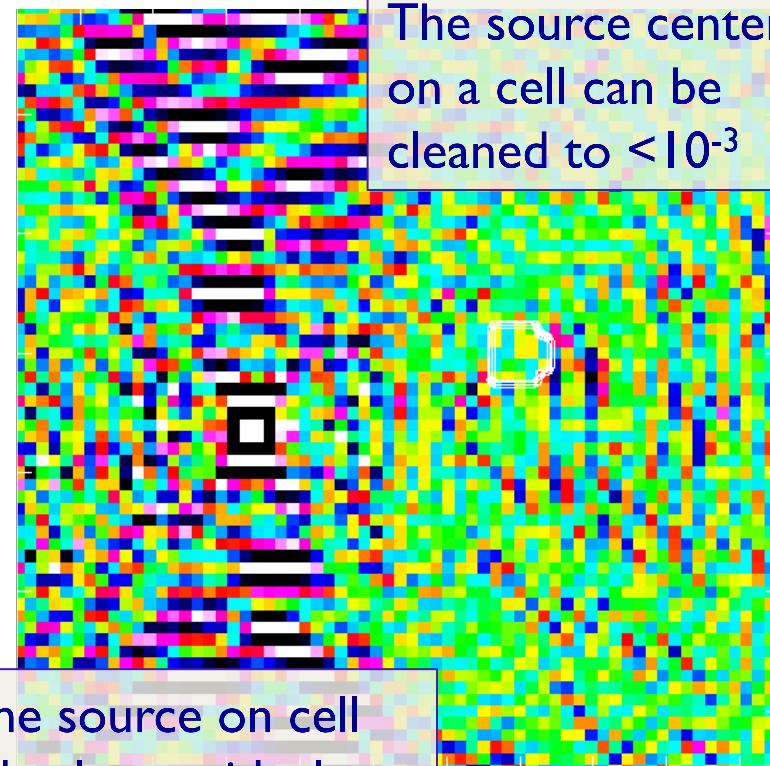


Why sample the PSF so finely?

- Gridding prevents arbitrary placement of model components

dirty image of 2 pt sources





The source centered on a cell can be cleaned to $<10^{-3}$

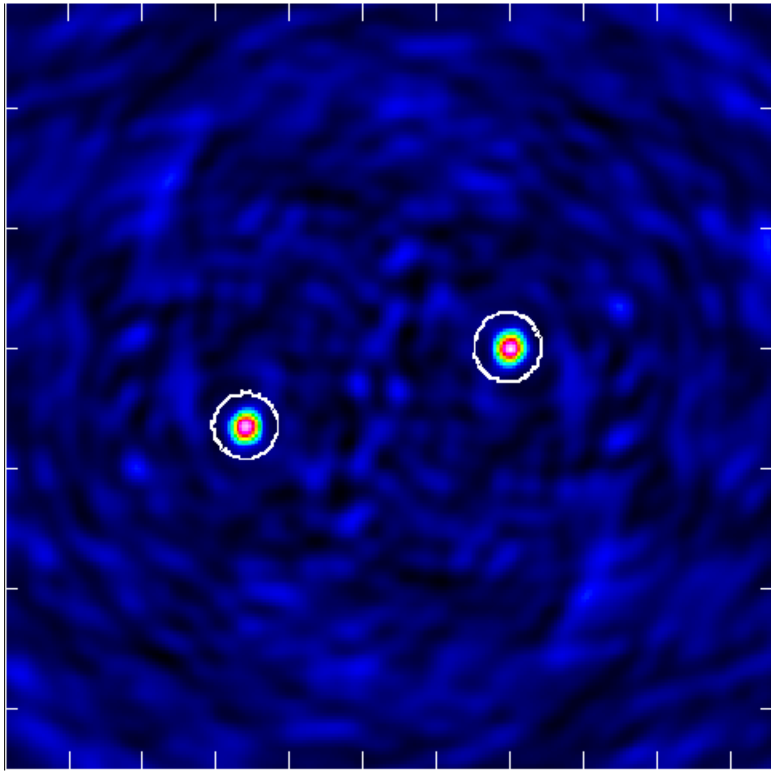
The source on cell edge has residuals at the $<10^{-2}$ level

linear scale ± 0.5%

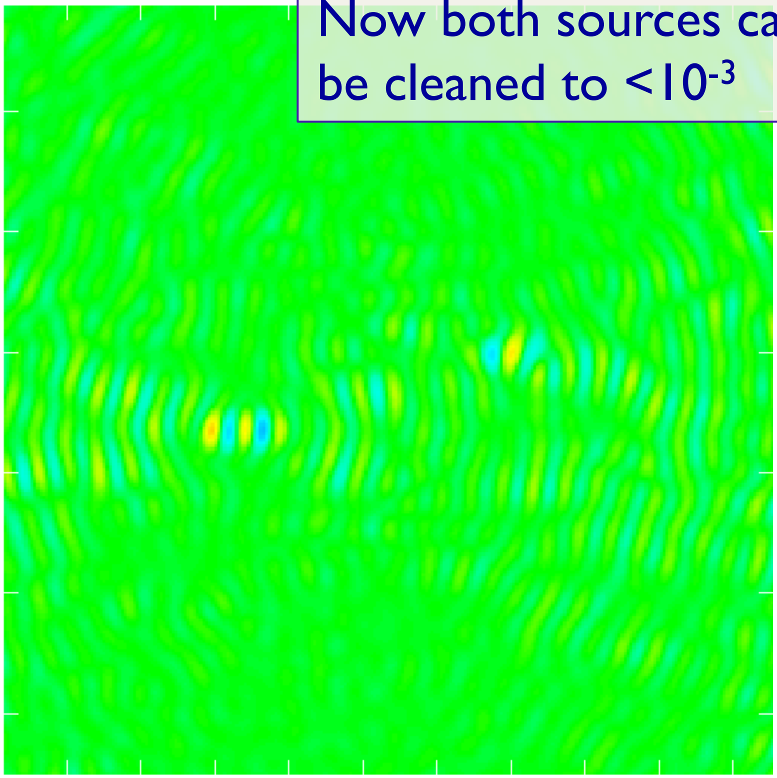
Why sample the PSF so finely?

- Gridding prevents arbitrary placement of model components

dirty image of 2 pt sources



Now both sources can be cleaned to $<10^{-3}$



linear scale ± 0.5%

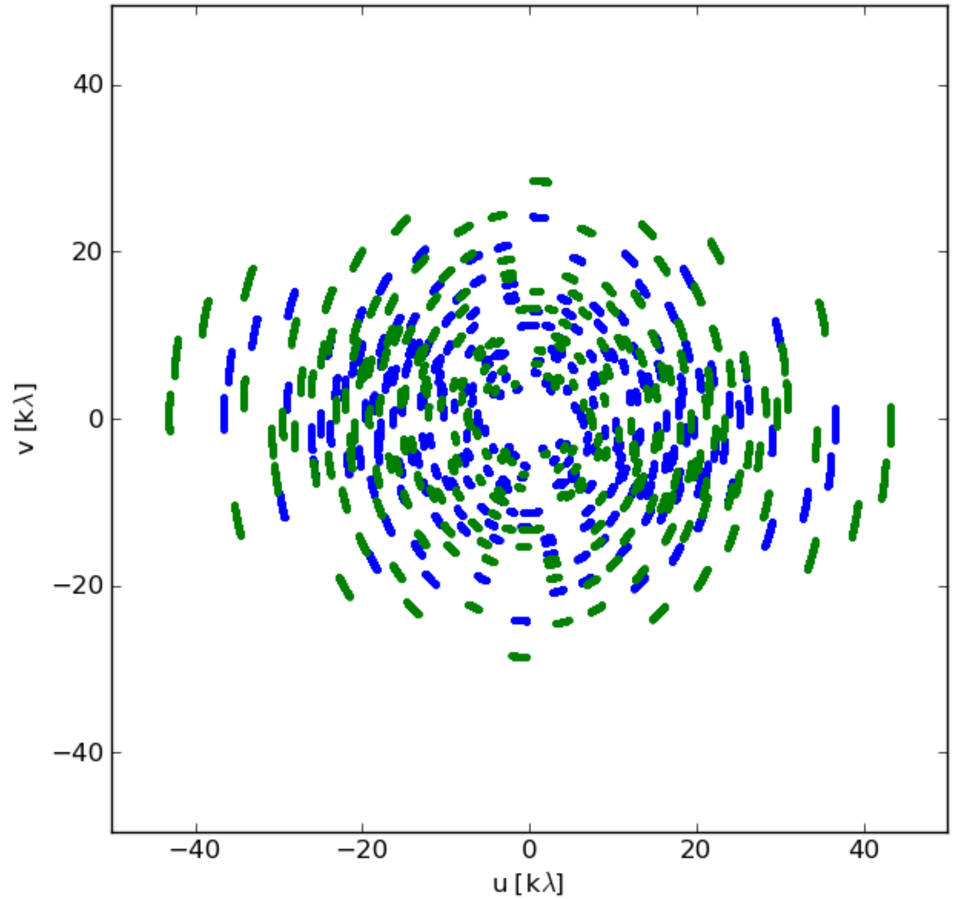
the clean task in casapy:

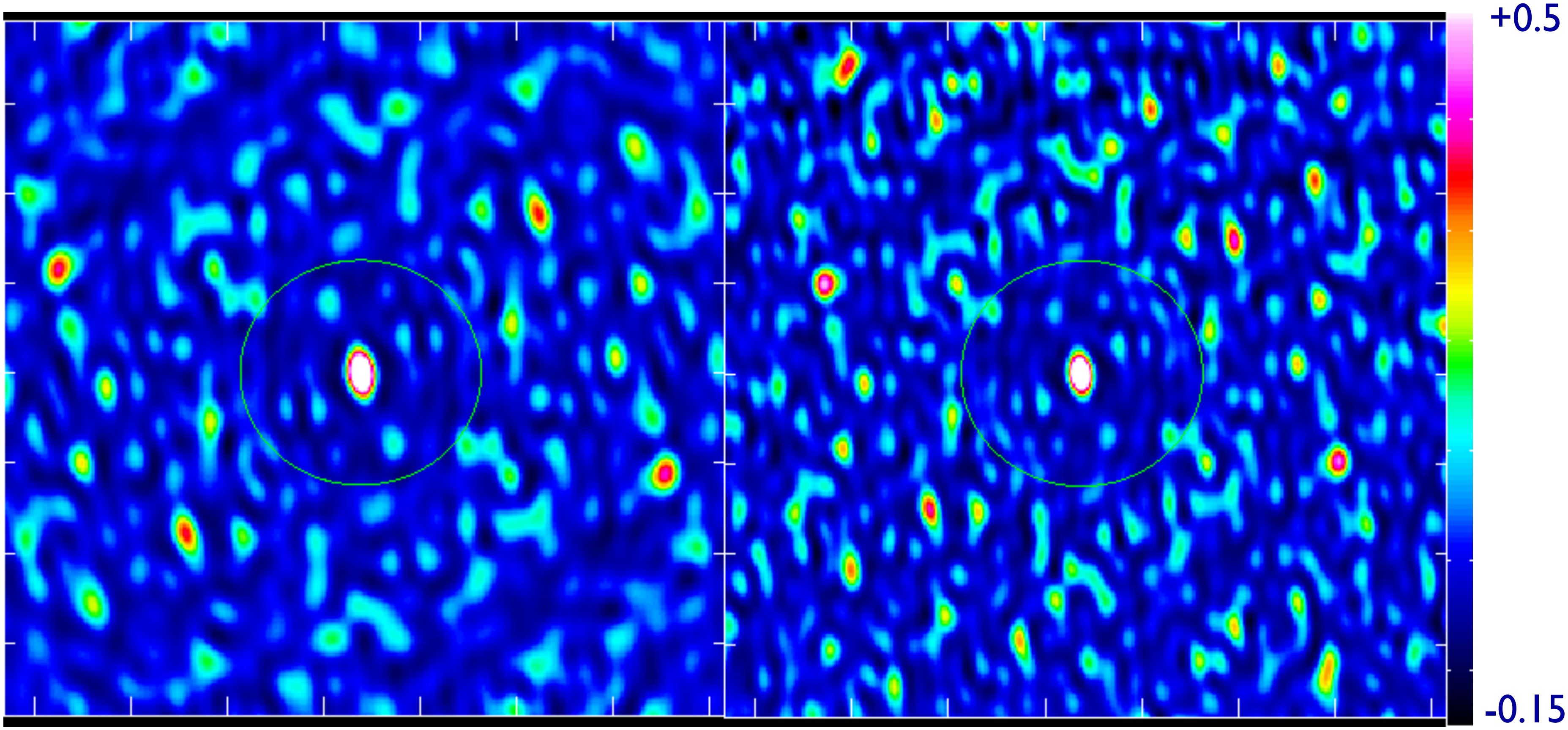
```
--------> inp(clean)
#  clean :: Invert and deconvolve images with selected algorithm
vis                 =         ''        #  Name of input visibility file
imagename           =         ''        #  Pre-name of output images
outlierfile         =         ''        #  Text file with image names, 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          =         ''        #  Options: 'csclean' or 'mosaic', '', uses psfmode
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
calready            =       True        #  True required for self-calibration
allowchunk          =      False        #  Divide large image cubes into channel chunks for deconvolution
async               =      False        #  If true the taskname must be started using clean(...)
```

- (visibility) data selection
- gridding & inversion
- deconvolution
- basic image parameters

Gridding & Inversion: MFS

```
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
```




+0.5
-0.15

higher order MFS

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

Average intensity at what frequency?

```
restfreq            =        ''        #  Rest frequency to assign to image (see help)
```

(stopping criteria also @ restfreq)

Gridding & Inversion: cubes

```
mode                = 'velocity'       #  Spectral gridding type (mfs, channel, velocity, frequency)
     nchan          =          -1      #  Number of channels (planes) in output image; -1 = all
     start          =           0      #  Velocity of first channel: e.g '0.0km/s'(''=first channel in first SpW of
                                       #   MS)
     width          =           1      #  Channel width e.g '-1.0km/s' (''=width of first channel in first SpW of MS)
     interpolation  =    'linear'      #  Spectral interpolation (nearest, linear, cubic). Use nearest for
                                       #   mode=channel
     chaniter       =       False      #  Clean each channel to completion (True), or all channels each cycle (False)
     outframe       =          ''      #  Velocity reference frame of output image; '' =input
     veltype        =     'radio'      #  Velocity definition of output image

phasecenter         = 'J2000 17:20:00.16300 -035.00.00.600000' #  Image center: direction or field index
restfreq            =          ''       #  Rest frequency to assign to image (see help)
```

mode = channel, velocity, frequency

- data: taken in sky frequency (terrestria,l TOPO) frame
  - velocity: include doppler shifts from:
    - Earth rotation: few km/s (diurnal)
    - Earth orbit: 30 km/s (annual)
    - Earth/Sun motion w.r.t. LSR (e.g. LSRK, LSRD)
    - maybe galactic rotation to extragalactic frames
- imaging applies doppler corrections on the fly, works in LSRK
- you choose the subsequent output cube parameters

- can shift and regrid data before imaging : cvel

Gridding & Inversion: mosaics

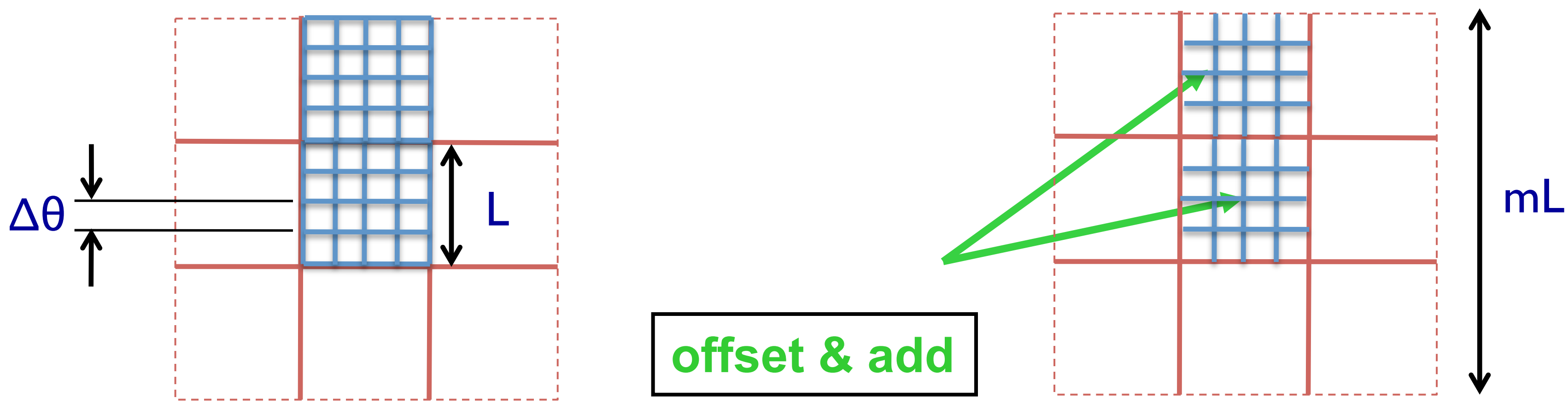
```
imagermode      =   'mosaic'   #  Options: 'csclean' or 'mosaic', '', uses psfmode
     mosweight  =      False   #  Individually weight the fields of the mosaic
     ftmachine  =       'ft'   #  Gridding method for the image
     scaletype  =    'SAULT'   #  Controls scaling of pixels in the image 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 = “ft” : shift and add in image plane

ftmachine = “mosaic” : add in uv plane and invert together

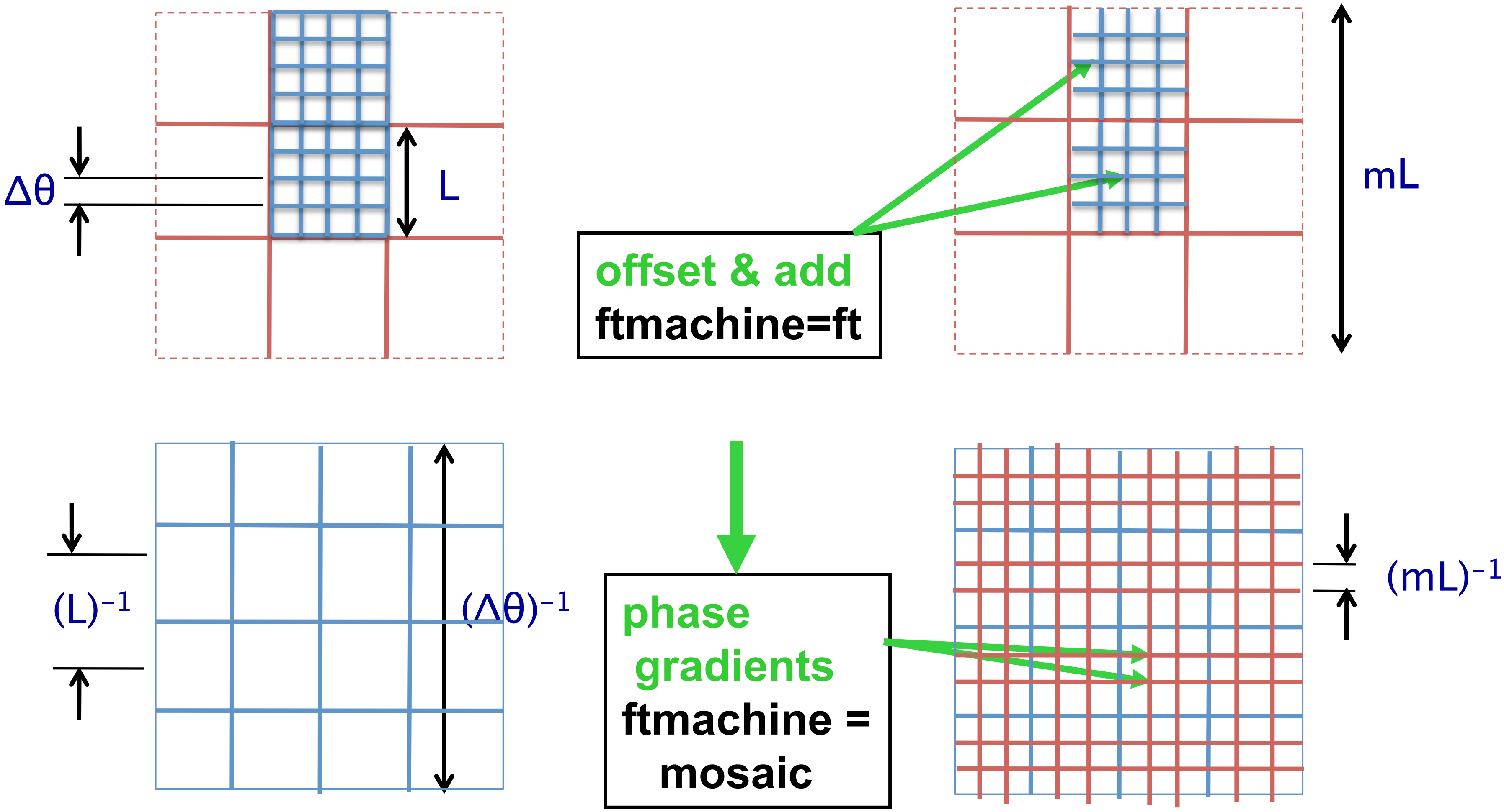
Gridding & Inversion: mosaics

Mosaicing in the sky plane



Gridding & Inversion: mosaics

Mosaicing in the sky and *uv* planes



FT{f(x)} = F(u) Fourier shift theorem $FT\{f(x-x_0)\} = F(u)e^{i2\pi x_0 u}$

Gridding & Inversion: mosaics

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

- A-projection kernel (FT of PB)
- uses approximate single PSF for entire mosaic
- uses POINTING table when present (FIELD for phases)

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

- slower
- sometimes better for sparsely sampled mosaic, and/or poor uv coverage

ftmachine = "" or "csclean" : single field

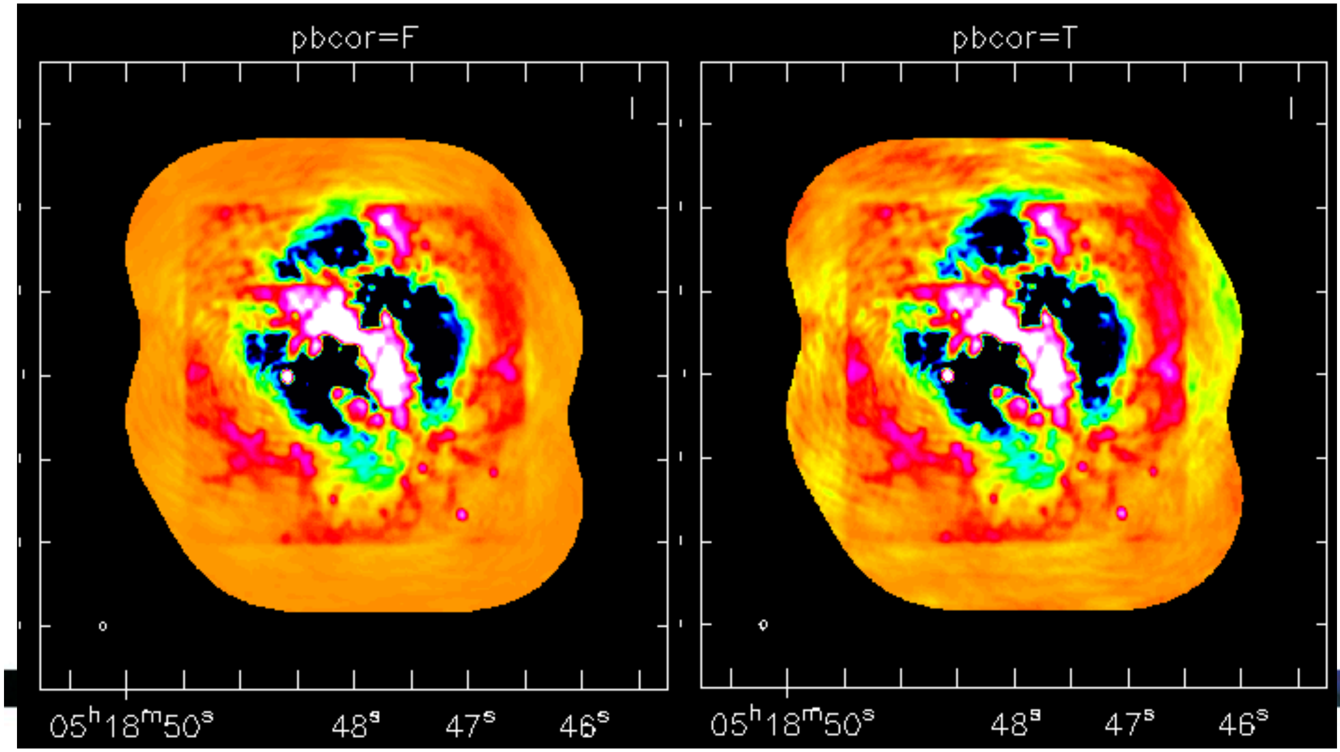
- spheroidal (highly tapered) gridding kernel

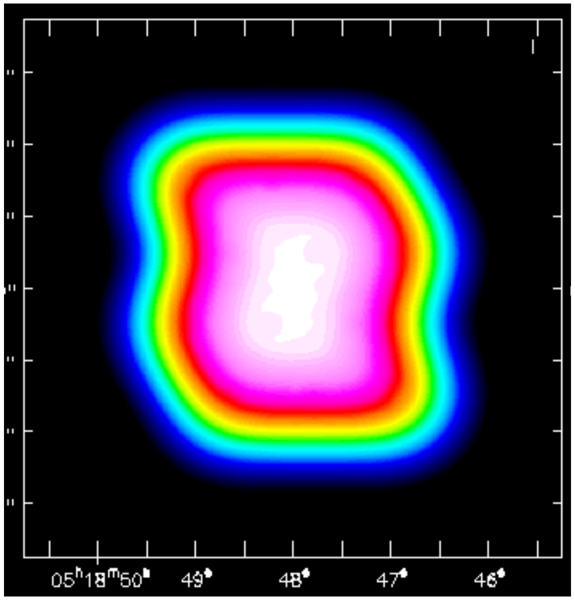
Gridding & Inversion: output

imagename.image
imagename.residual
imagename.model
imagename.psf

```
pbcor                =       False        #  Output primary beam-corrected image
minpb                =         0.2        #  Minimum PB level to use
```

imagename.flux (PB plus weights plus extra PB from A-convolution)
imagename.flux.pbcoverage (just the PB effects)
to correct to flux on-sky use .image/.flux



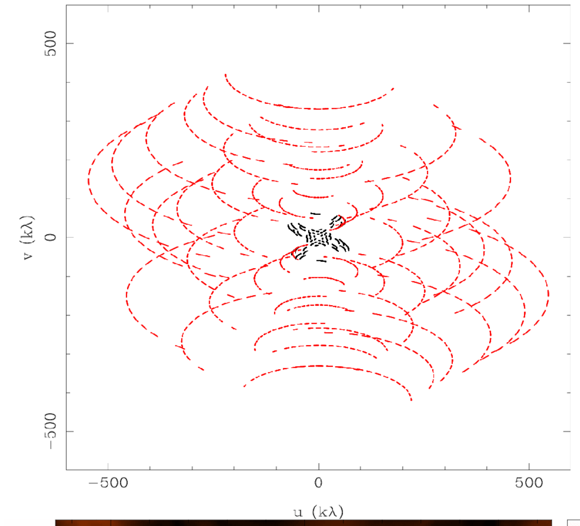


Dirty Beam Shape and Weighting

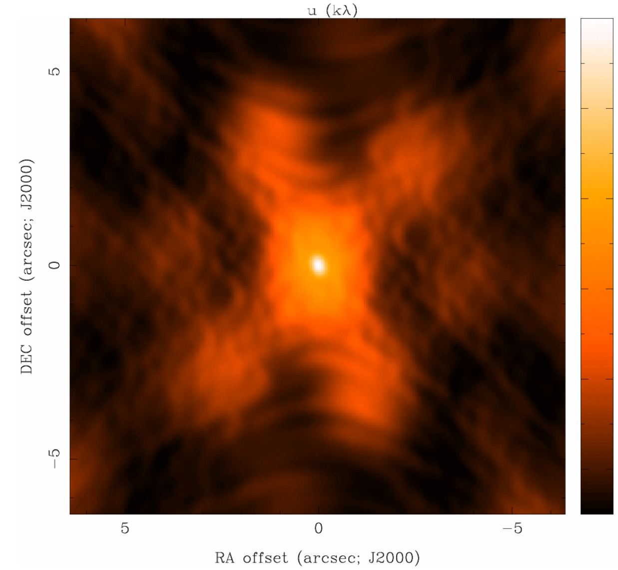
- introduce weighting function W(u,v)

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

  – W modifies sidelobes of dirty beam (W is also gridded for FFT)

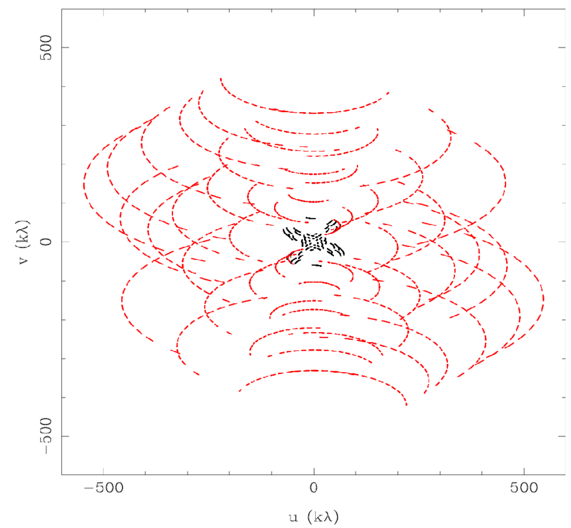


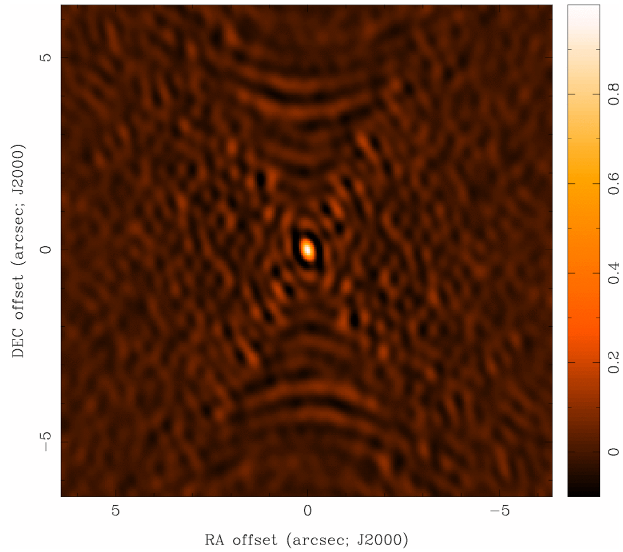
- "Natural" weighting
  – $W(u,v) = 1/\sigma^2(u,v)$ at points with data and zero elsewhere, where $\sigma^2(u,v)$ is the noise variance of the (u,v) sample
  – maximizes point source sensitivity (lowest rms in image)
  – generally more weight to short baselines (large spatial scales), degrades resolution



Dirty Beam Shape and Weighting

- "Uniform" weighting
  - W(u,v) is inversely proportional to local density of (u,v) points, so sum of weights in a (u,v) cell is a constant (or zero)
  - fills (u,v) plane more uniformly, so (outer) sidelobes are lower
  - gives more weight to long baselines and therefore higher angular resolution
  - degrades point source sensitivity (higher rms in image)
  - can be trouble with sparse sampling: cells with few data points have same weight as cells with many data points



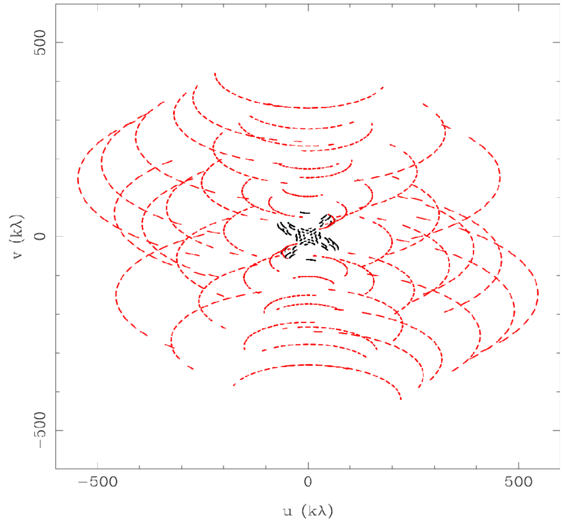


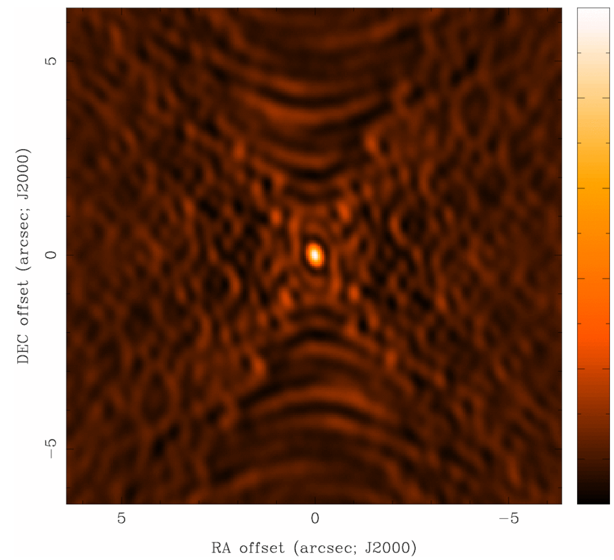
Dirty Beam Shape and Weighting

- "Robust" (Briggs) weighting
  - variant of "uniform" that avoids giving too much weight to cell with low natural weight
  - implementations differ, e.g. S_N is natural weight of a cell, S_t is a threshold

$$W(u,v) = \frac{1}{\sqrt{1+S_N^2/S_{thresh}^2}}$$

  - large threshold → natural weighting
  - small threshold → uniform weighting
  - an adjustable parameter that allows for continuous variation between highest angular resolution and optimal point source sensitivity





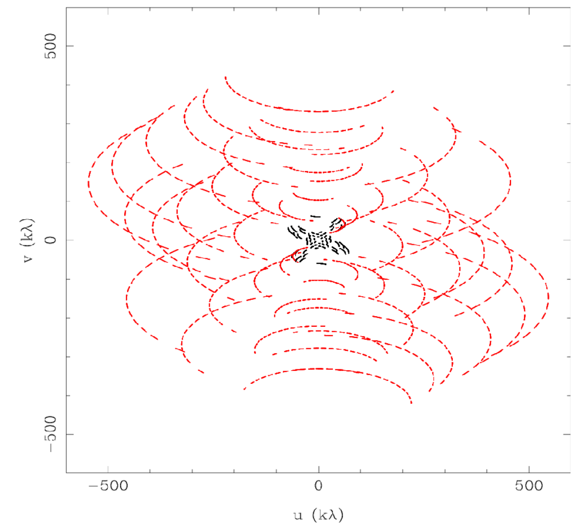
Dirty Beam Shape and Weighting

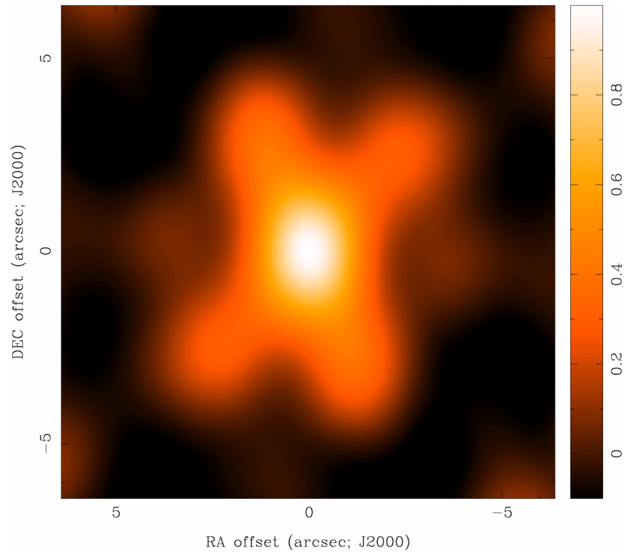
- "Tapering"
  - apodize the (u,v) sampling by a Gaussian

$$W(u,v) = exp\left\{-\frac{(u^2+v^2)}{t^2}\right\}$$

t = tapering parameter (in kλ; arcsec)

  - like smoothing in the image plane (convolution by a Gaussian)
  - gives more weight to short baselines, degrades angular resolution
  - degrades point source sensitivity but can improve sensitivity to extended structure



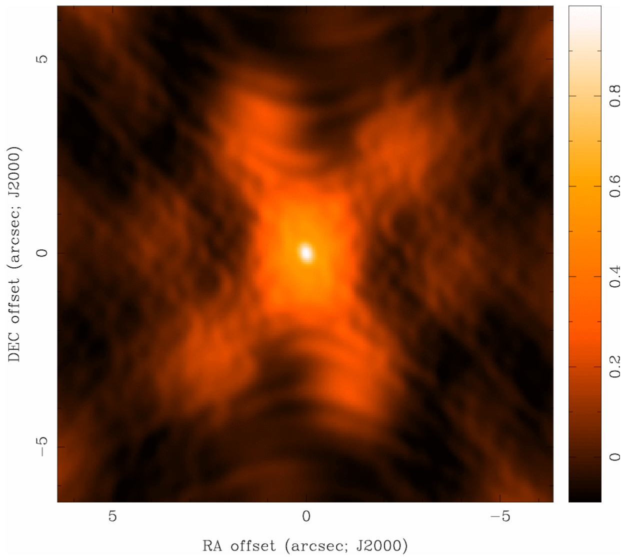


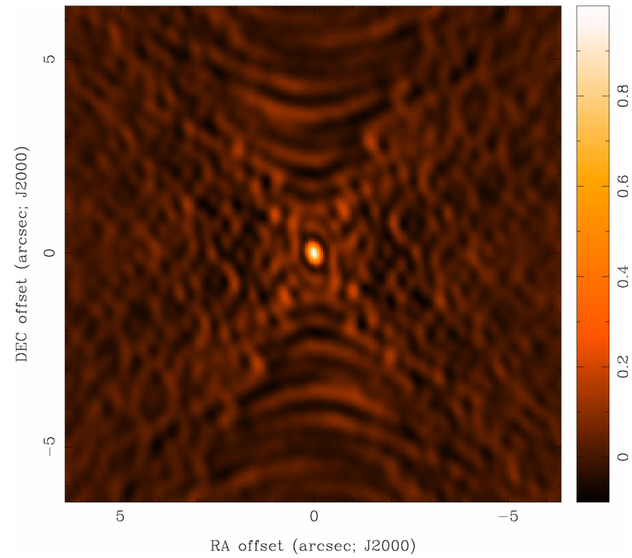
NRAO

Weighting and Tapering: Noise

Natural
0.77x0.62

σ=1.0



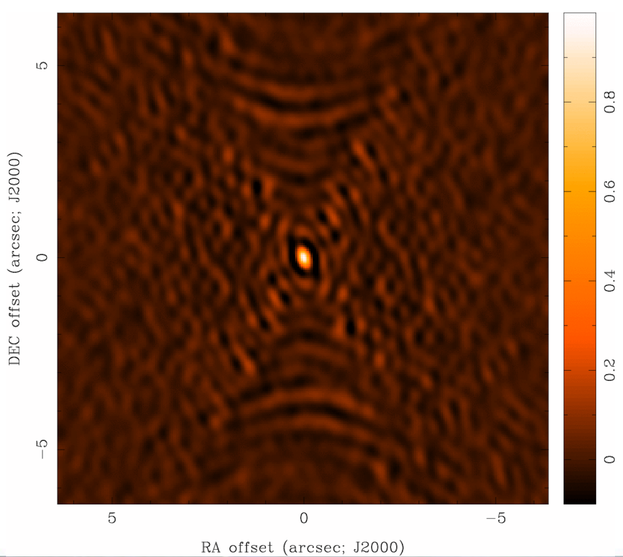


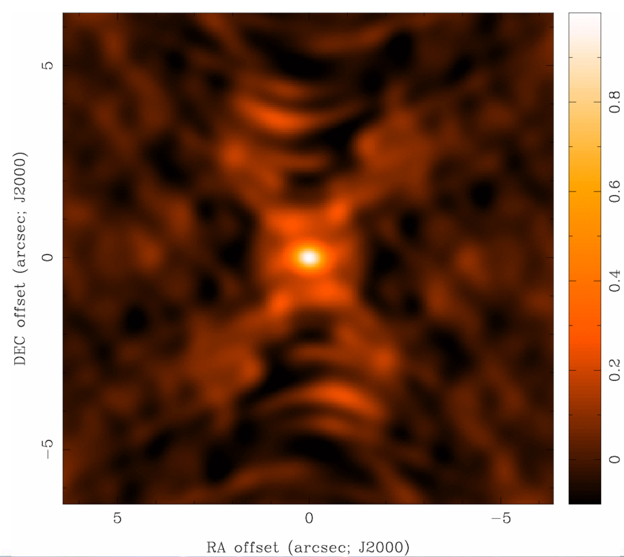
Robust 0
0.41x0.36

σ=1.6

Uniform
0.39x0.31

σ=3.7





Robust 0
+ Taper
0.77x0.62

σ=1.7

Weighting and Tapering: Summary

- imaging parameters provide a lot of freedom
- appropriate choice depends on science goals

	Robust/Uniform	Natural	Taper
Resolution	higher	medium	lower
Sidelobes	lower	higher	depends
Point Source Sensitivity	lower	maximum	lower
Extended Source Sensitivity	lower	medium	higher

the clean task in casapy:

- (visibility) data selection
- gridding & inversion
- deconvolution
- basic image parameters

```
--------> 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          =         ''        #  Options: 'csclean' or 'mosaic', '', uses psfmode
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
calready            =       True        #  True required for self-calibration
allowchunk          =      False        #  Divide large image cubes into channel chunks for deconvolution
async               =      False        #  If true the taskname must be started using clean(...)
```

Deconvolution algorithms

data visibilities

grid & FFT

dirty image

Iterative removal of
dirty beam

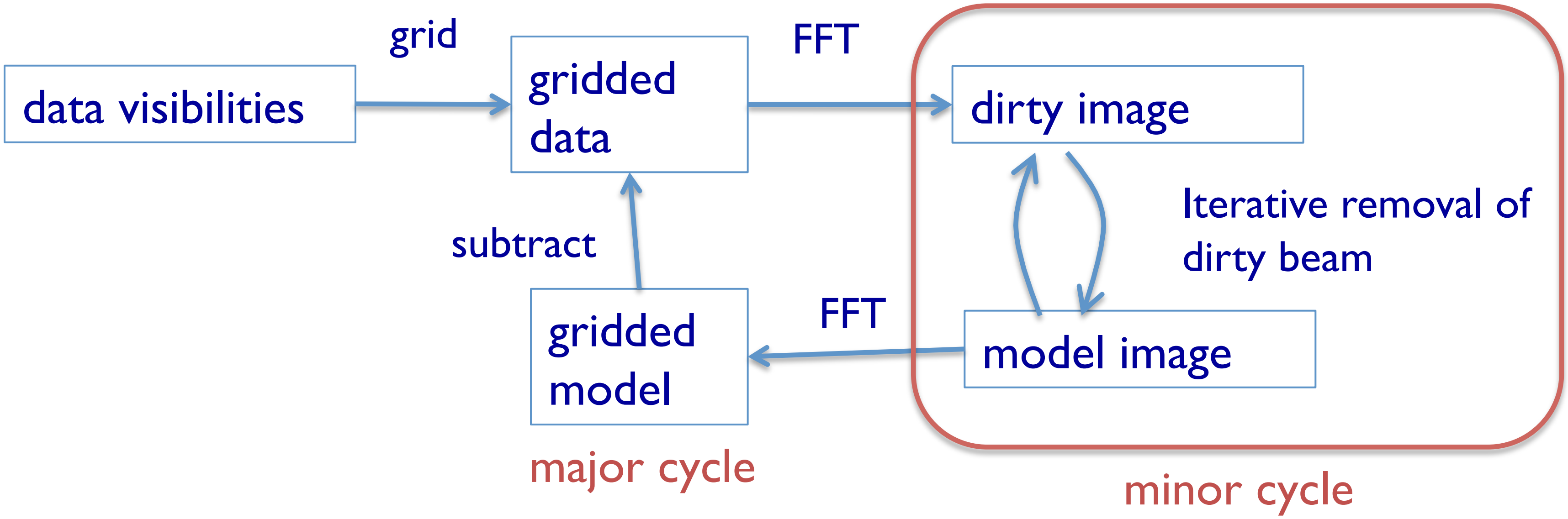
model image

Hogbom clean: (Högbom 1974)

subtracts full PSF in image domain

fast but not very accurate: errors build

Deconvolution algorithms



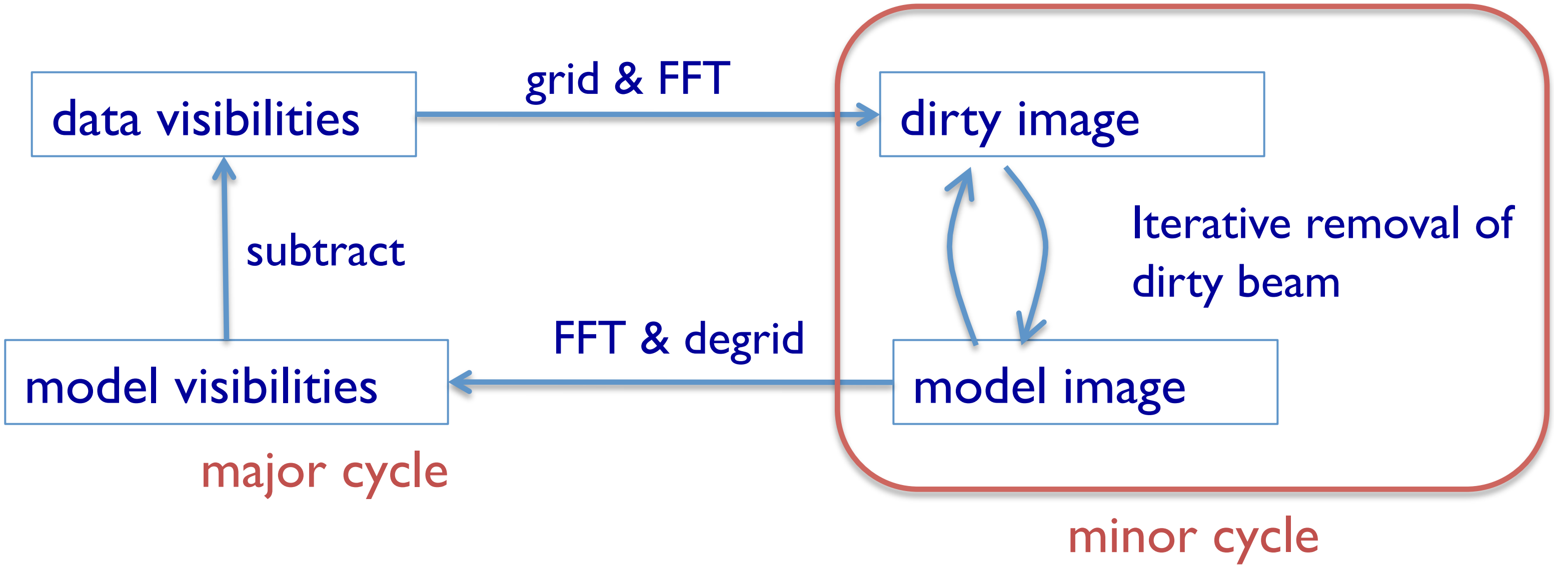
Clark :

subtracts truncated PSF in image domain

periodically subtracts from gridded data in uv domain

major/minor cycle frequency controlled by cyclefactor parameter

Deconvolution algorithms



Cotton-Schwab :

subtracts truncated PSF in image domain

major cycle subtracts from full visibilities

significant I/O per major cycle

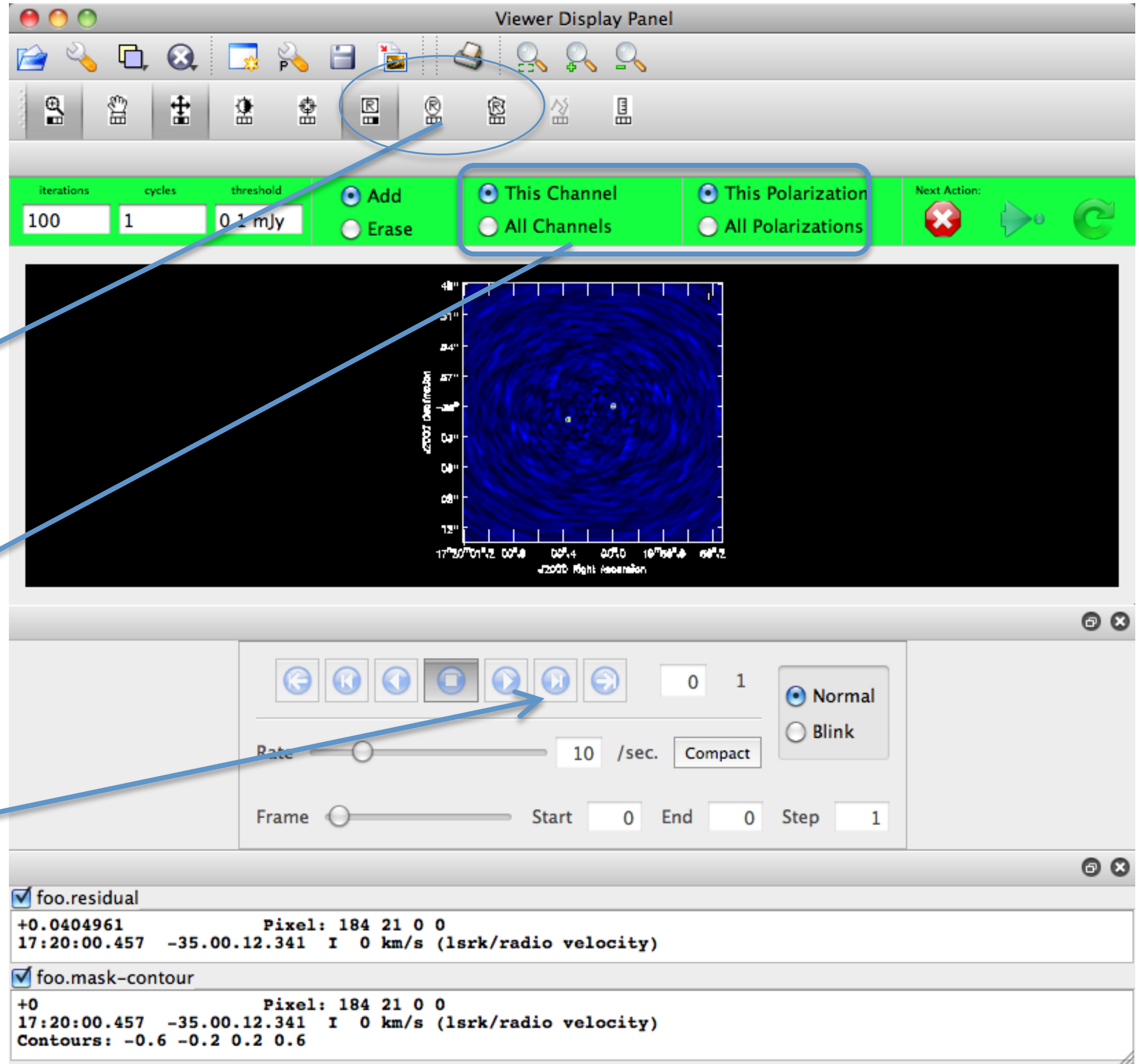
Stopping criteria

```
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'
```

- # iterations: arbitrary – set large and only use for safety
- threshold: max of residual map
  - multiple of rms noise if noise-limited
  - fraction of brightest source peak flux if dynamic-range limited

Interactive Clean

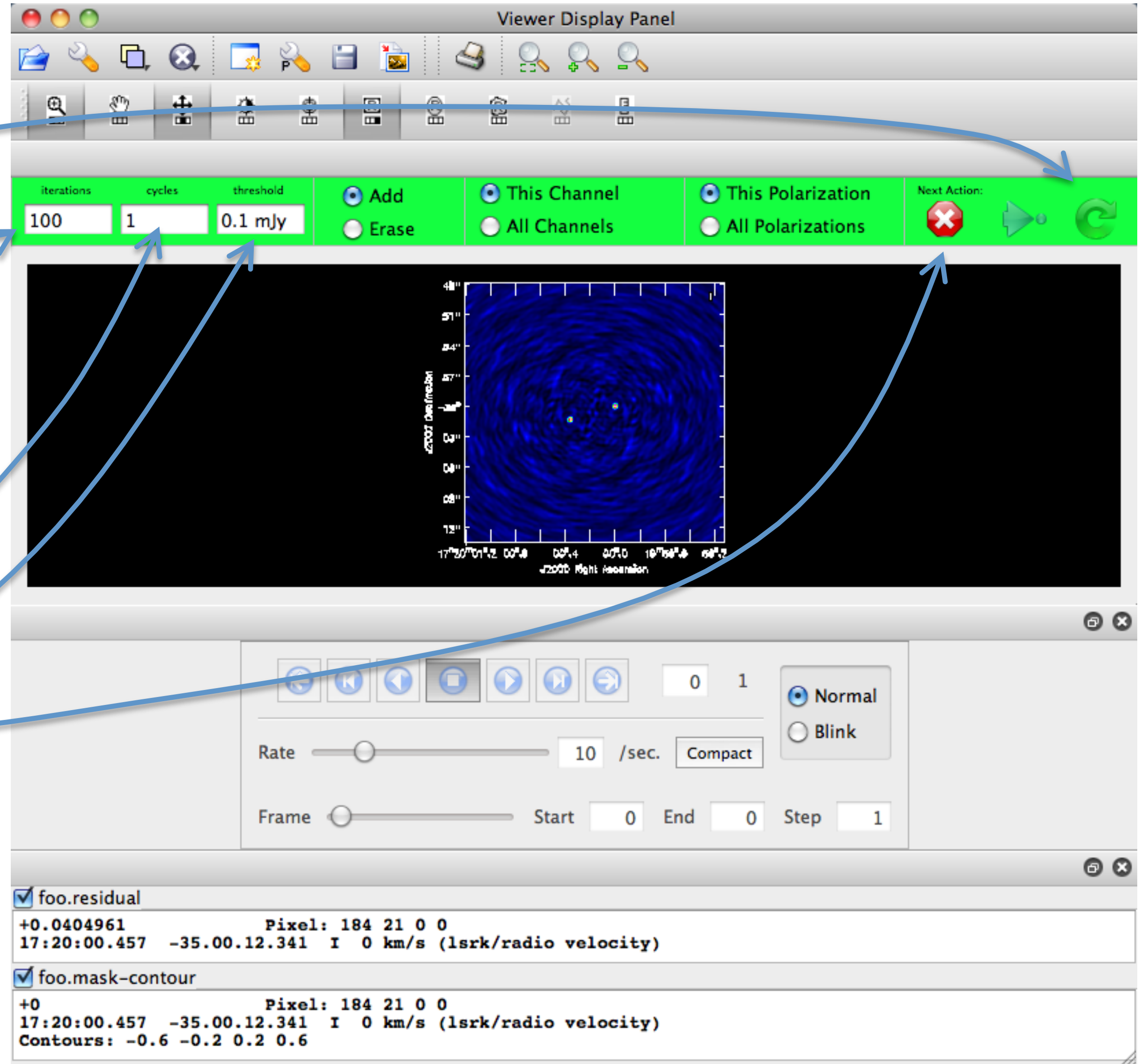
- residual image in viewer
- define a mask with R-click on shape type
- define the same mask for all channels
- or iterate through the channels with the tape deck and define separate masks



NRAO

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



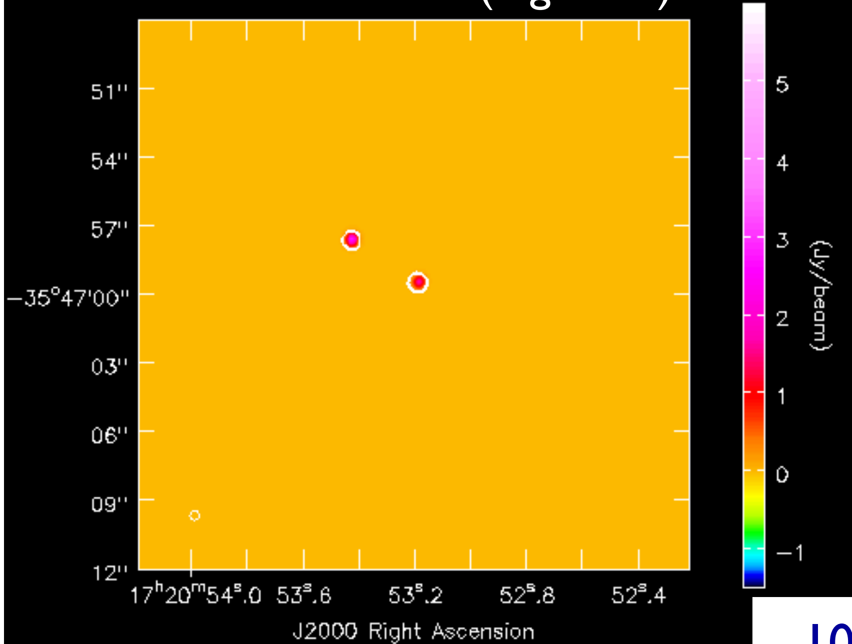
Sparse Approximation Imaging [multiscale]

- Problem: find a model to represent the sky as efficiently as possible, subject to the data constraints and within the noise uncertainty, possibly also subject to prior constraints.
  - some problems (like ours) cannot be efficiently reconstructed using orthonormal bases (like pixels or Fourier modes)
  - use non-orthogonal bases: multiscale (e.g. Gaussians)
  - choose *dictionary* of model elements (*atoms*)
  - efficiency: find a representation that uses the fewest number of atoms

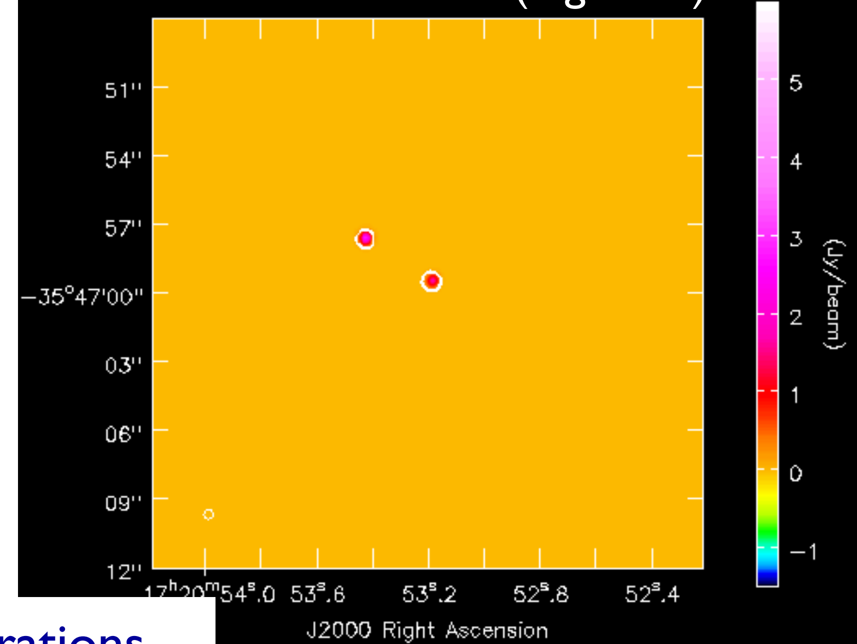
- Multiscale = [0,5,15,45]
  - scales are in units of pixels
  - 0=point, typically 1-2x synthesized beam, then multiples of 2-3x that up to a fraction of the PB can be tricky to get to work right

  - separate clean multiscale components not kept (Minor Cycle algorithm)

1-scale model (log scale)

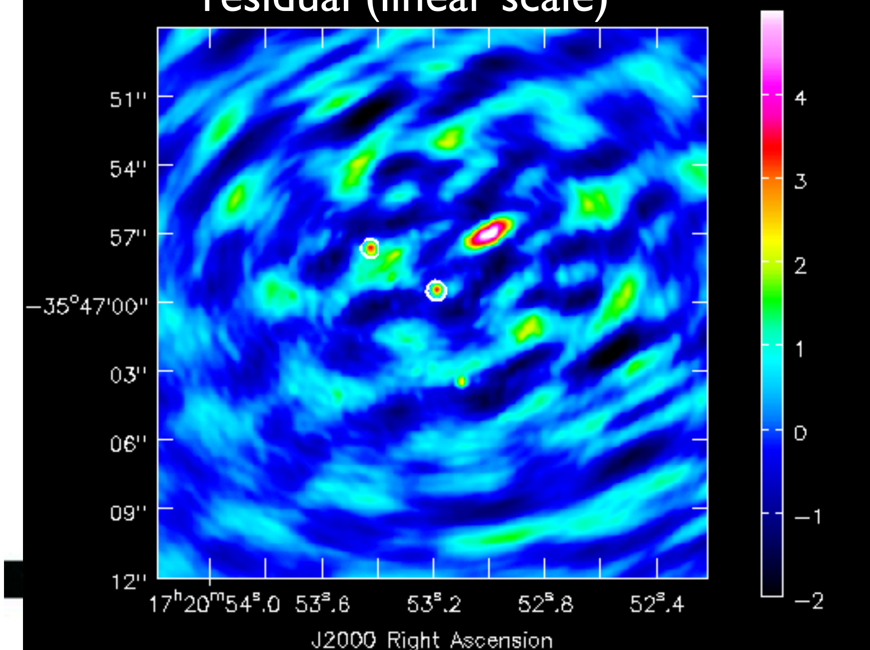


multiscale model (log scale)

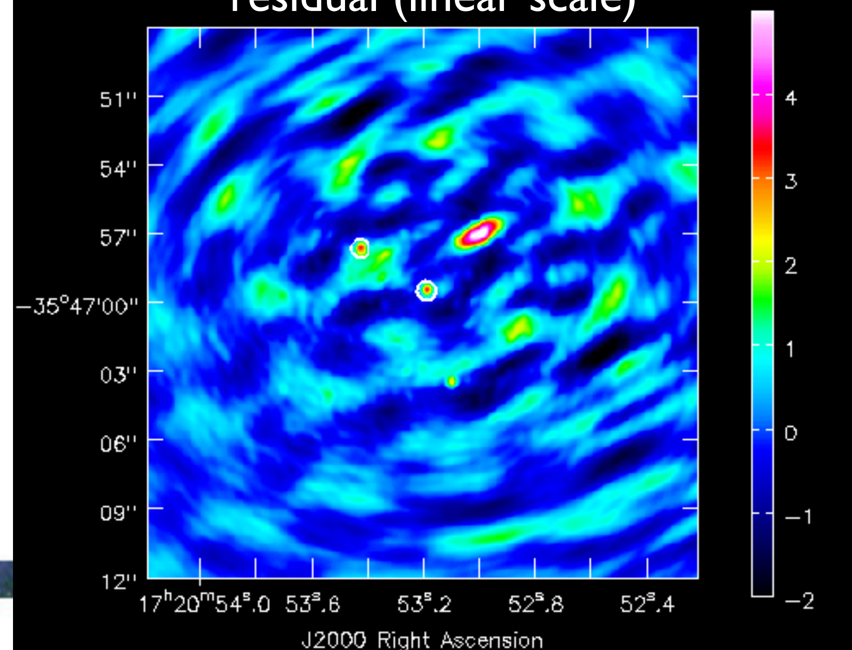


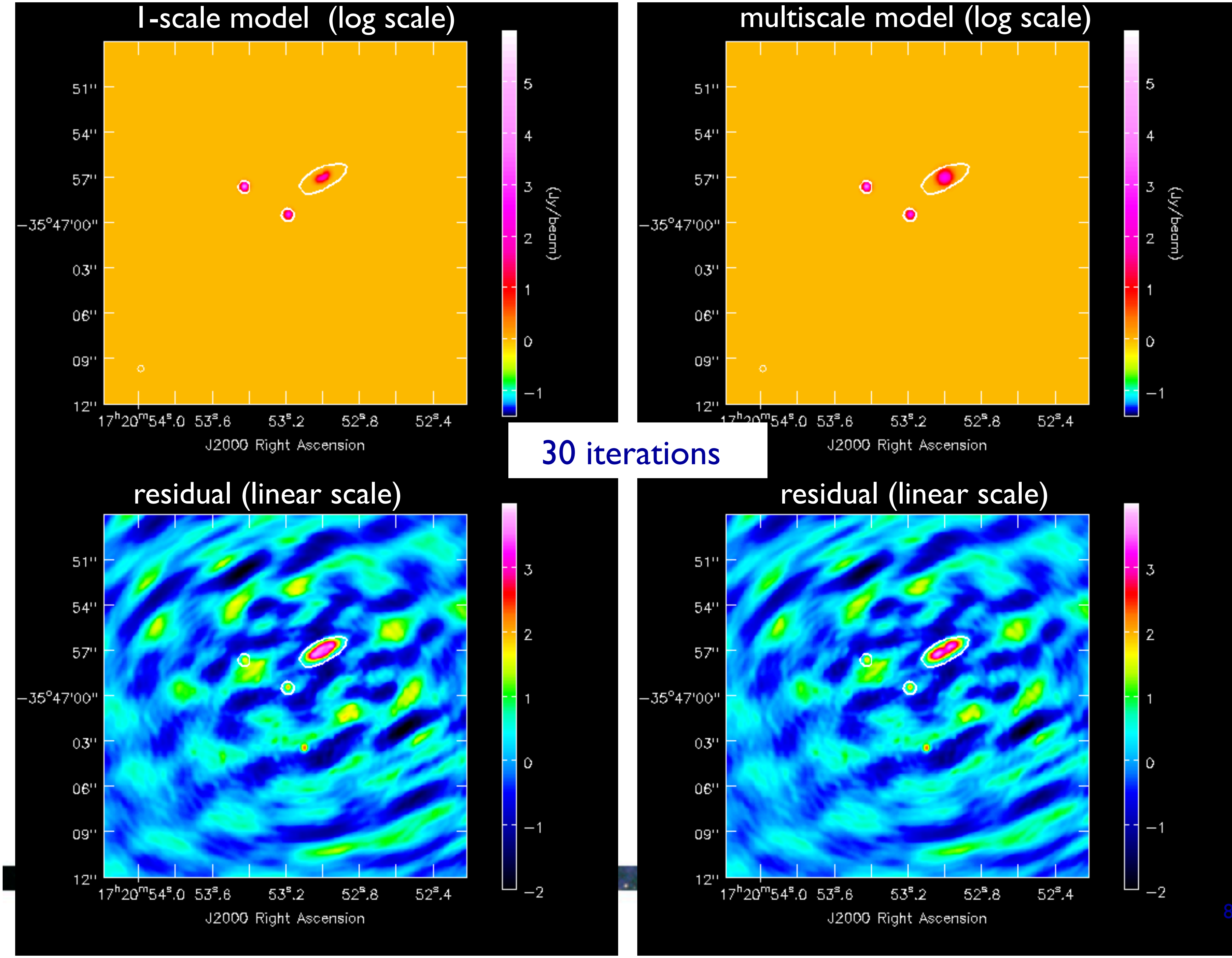
10 iterations

residual (linear scale)

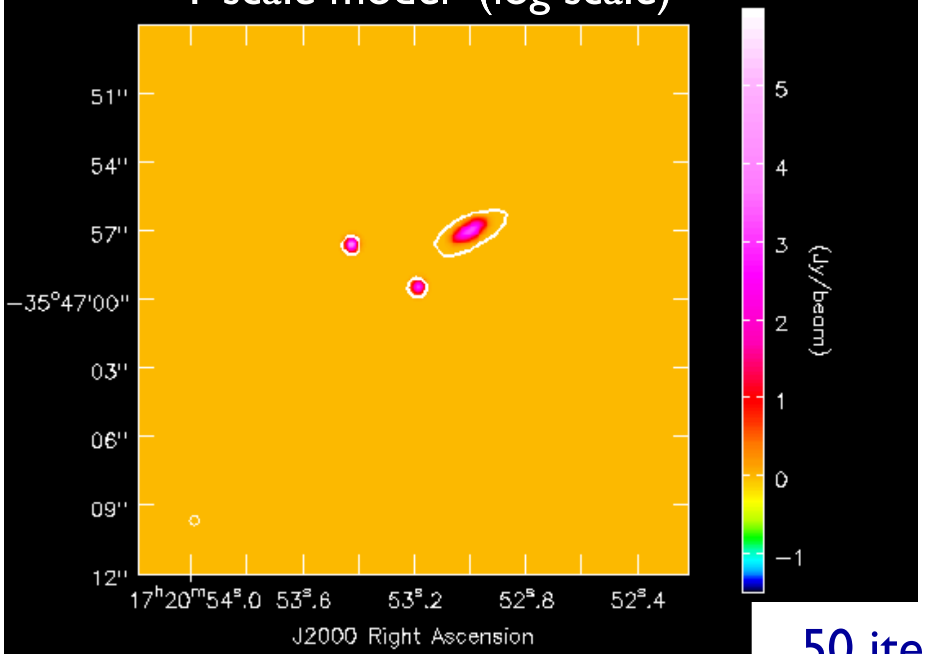


residual (linear scale)

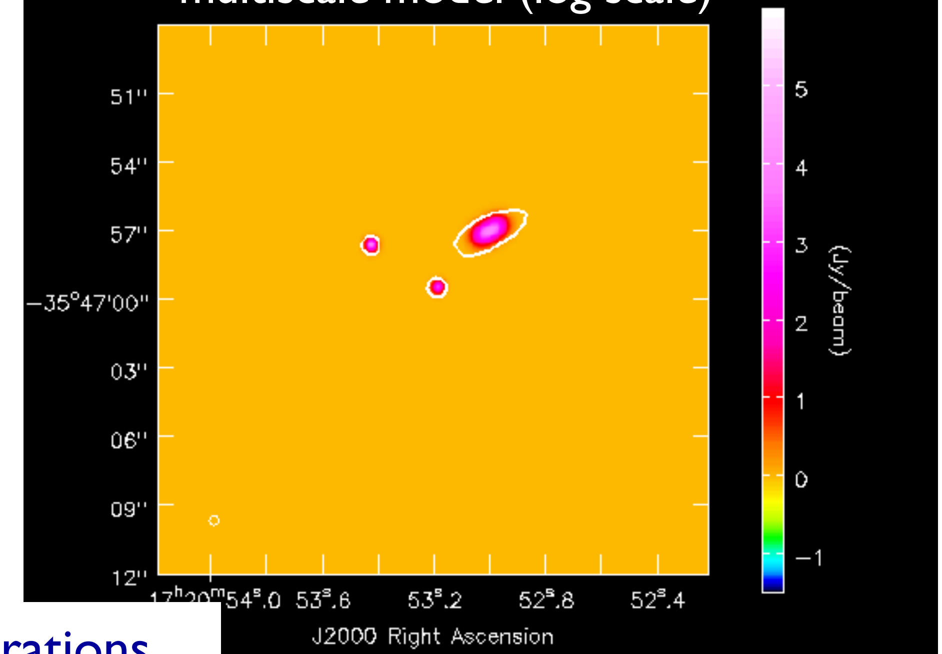



1-scale model (log scale)
multiscale model (log scale)
residual (linear scale)
residual (linear scale)
30 iterations
J2000 Right Ascension
(Jy/beam)
51''
54''
57''
−35°47'00''
03''
06''
09''
12''
17h20m54s.0
53s.6
53s.2
52s.8
52s.4

1-scale model (log scale)

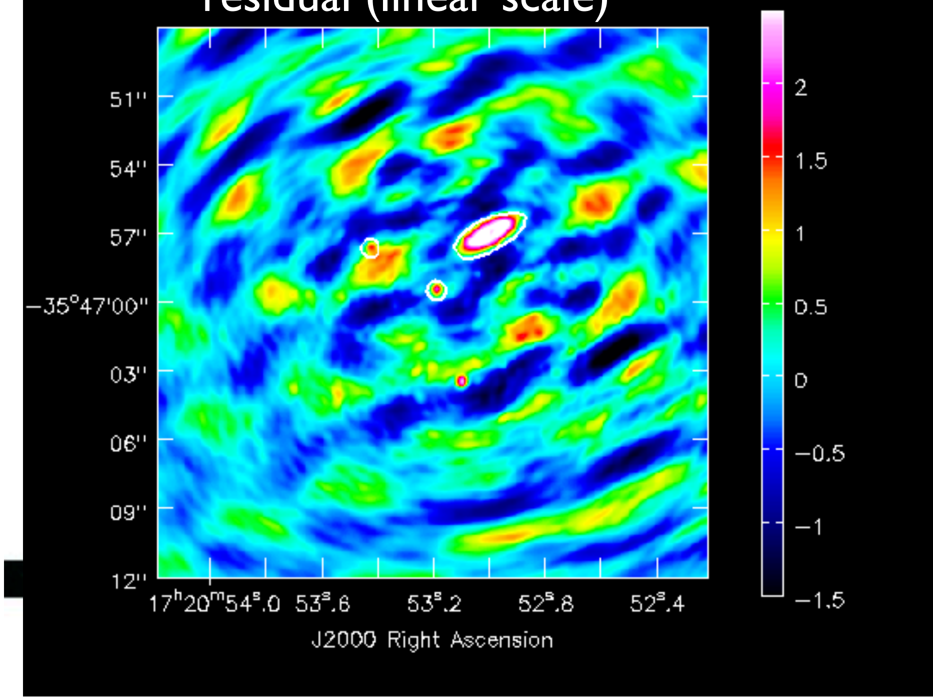


multiscale model (log scale)

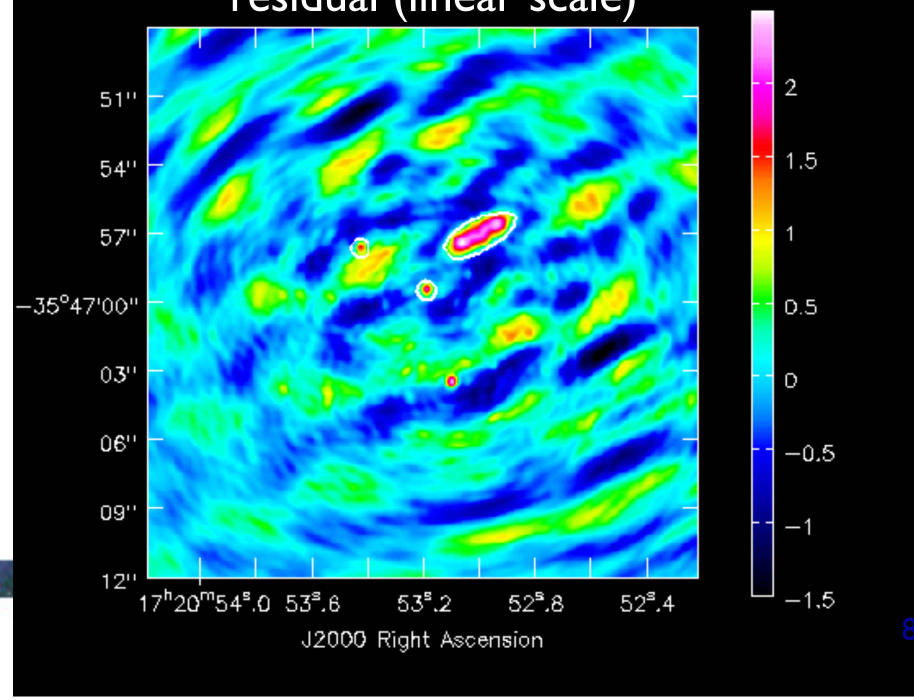


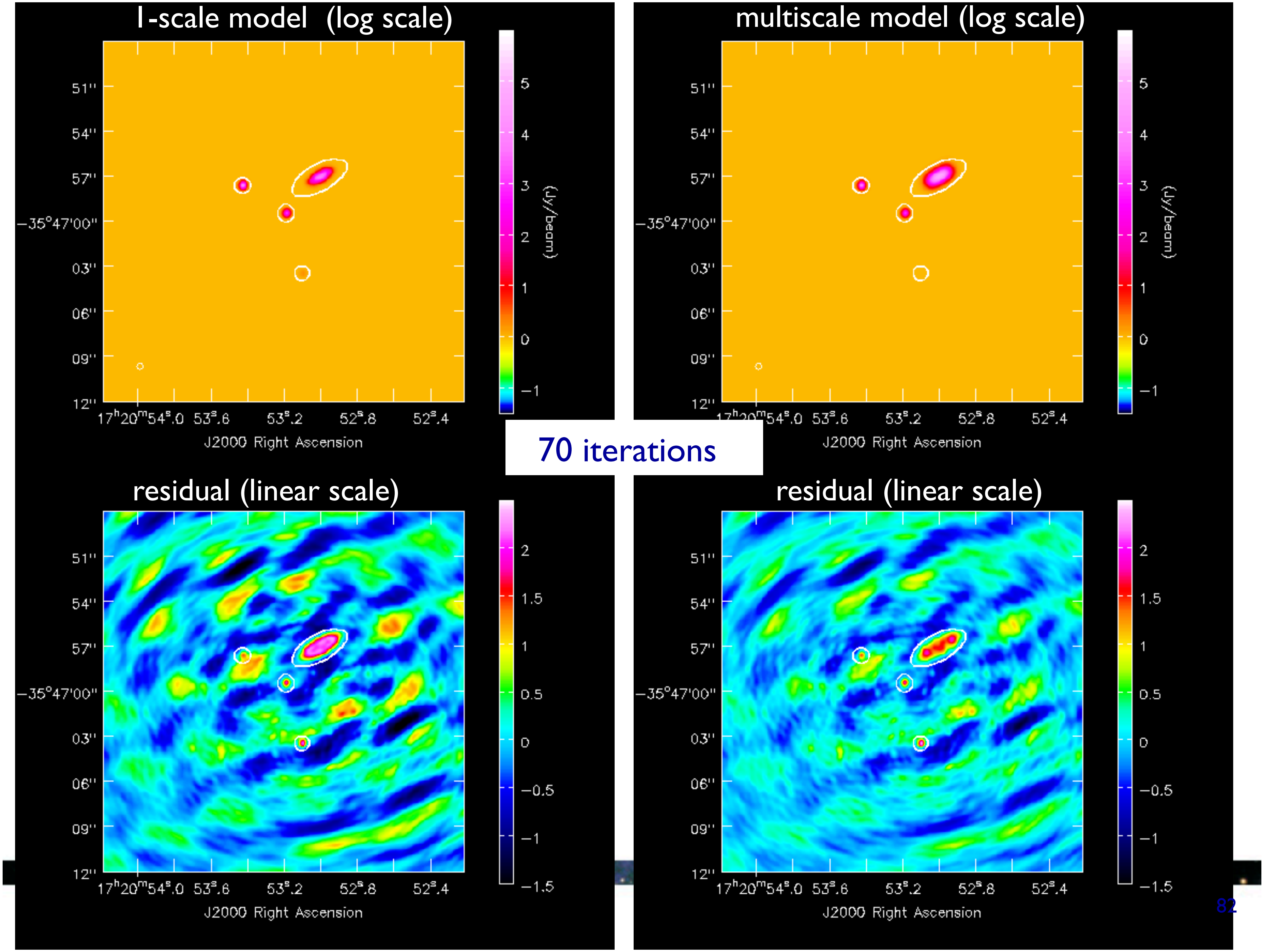
50 iterations

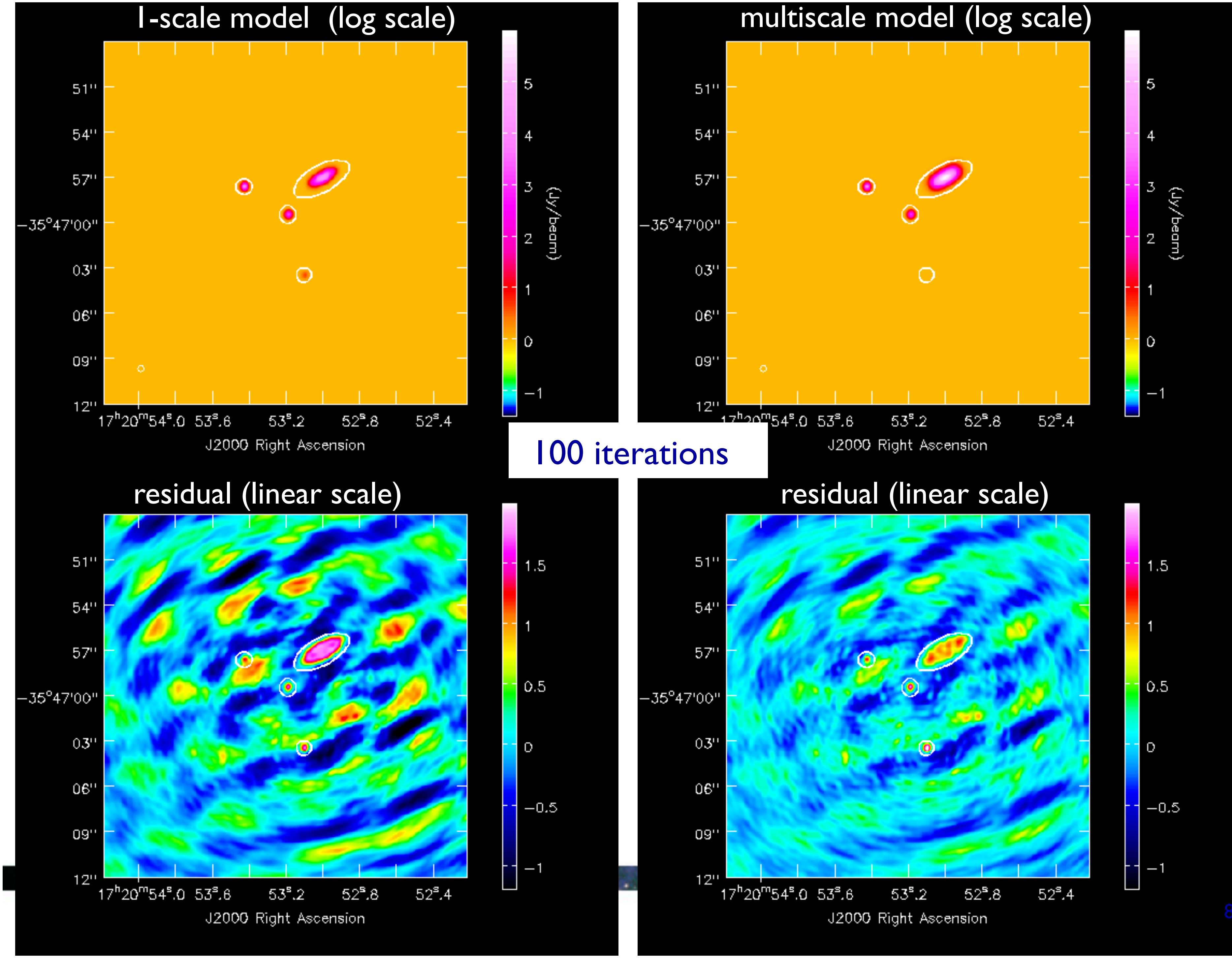
residual (linear scale)



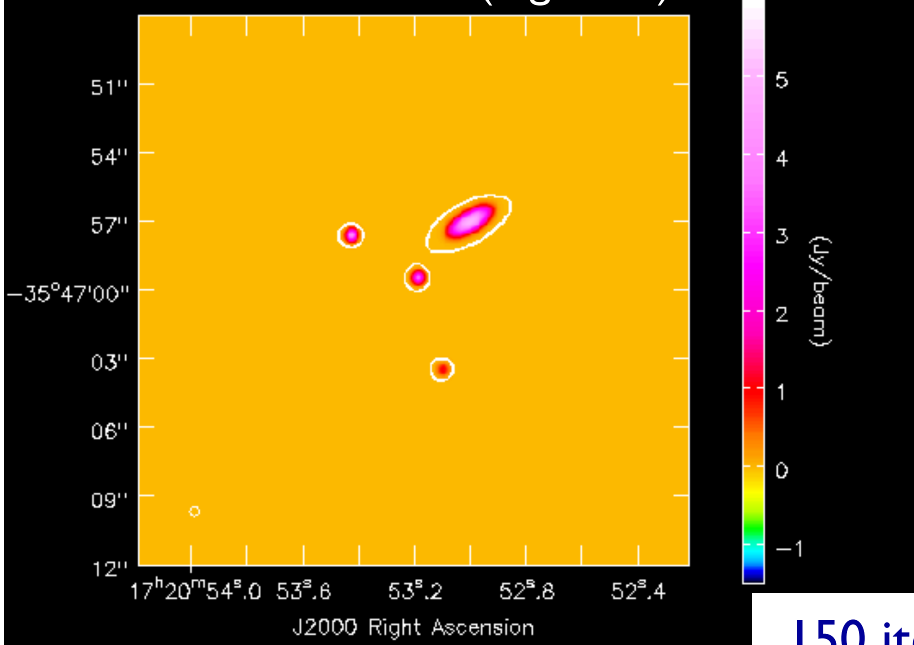
residual (linear scale)



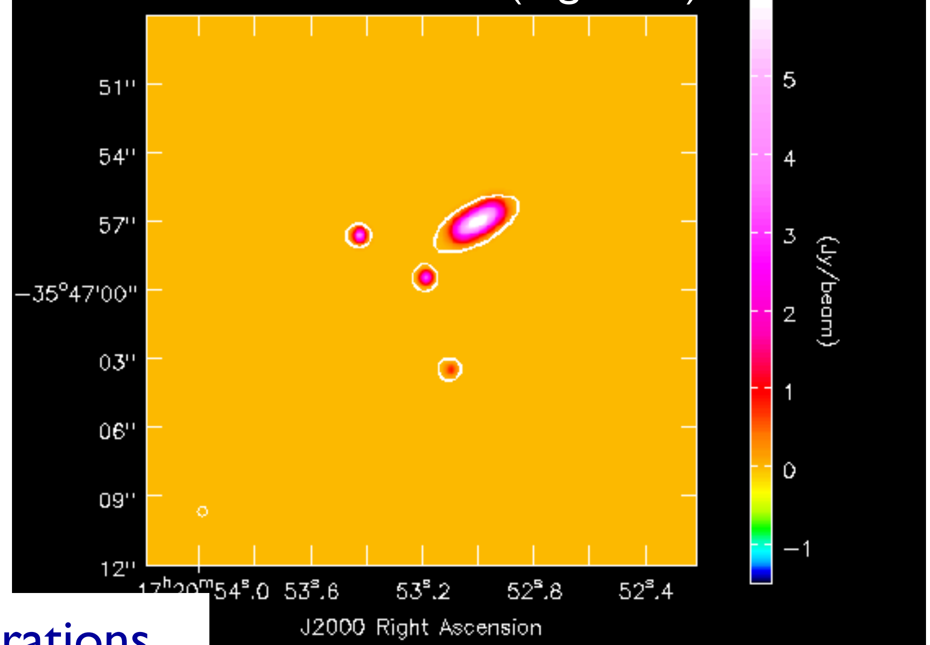

1-scale model (log scale)
multiscale model (log scale)
70 iterations
residual (linear scale)
residual (linear scale)
J2000 Right Ascension
(Jy/beam)


1-scale model (log scale)
multiscale model (log scale)
100 iterations
residual (linear scale)
residual (linear scale)
J2000 Right Ascension
(Jy/beam)
17h20m54s.0 53s.6 53s.2 52s.8 52s.4
51'' 54'' 57'' −35°47'00'' 03'' 06'' 09'' 12''

1-scale model (log scale)

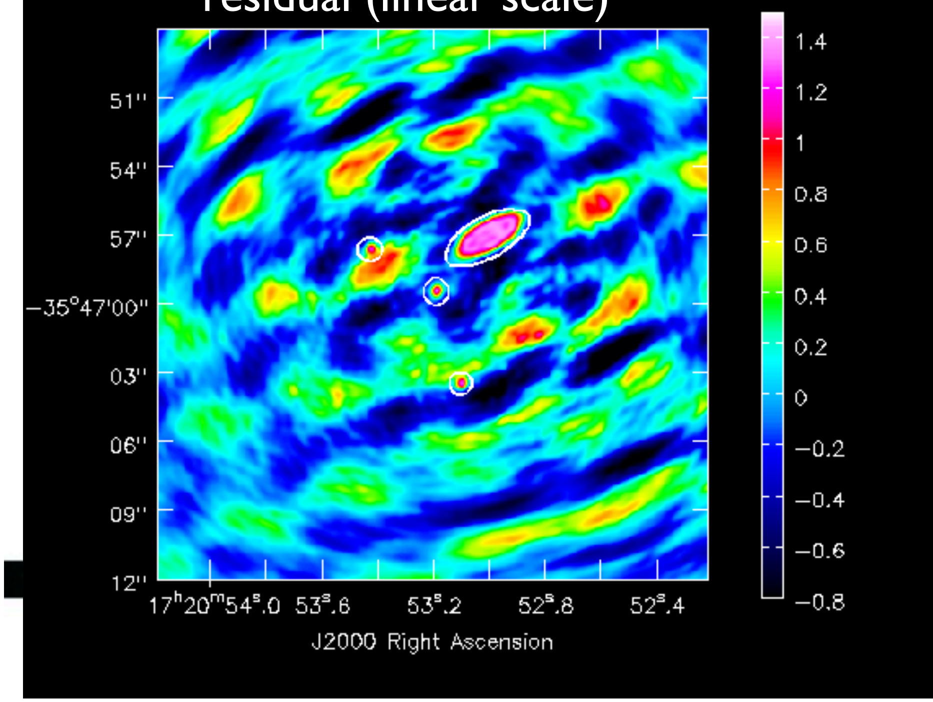


multiscale model (log scale)

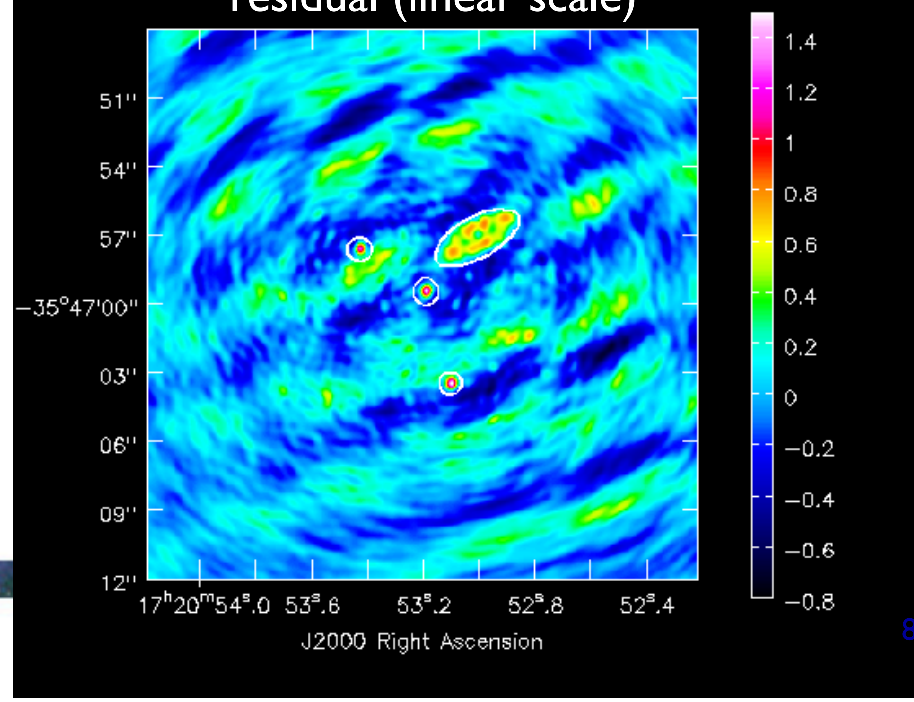


150 iterations

residual (linear scale)



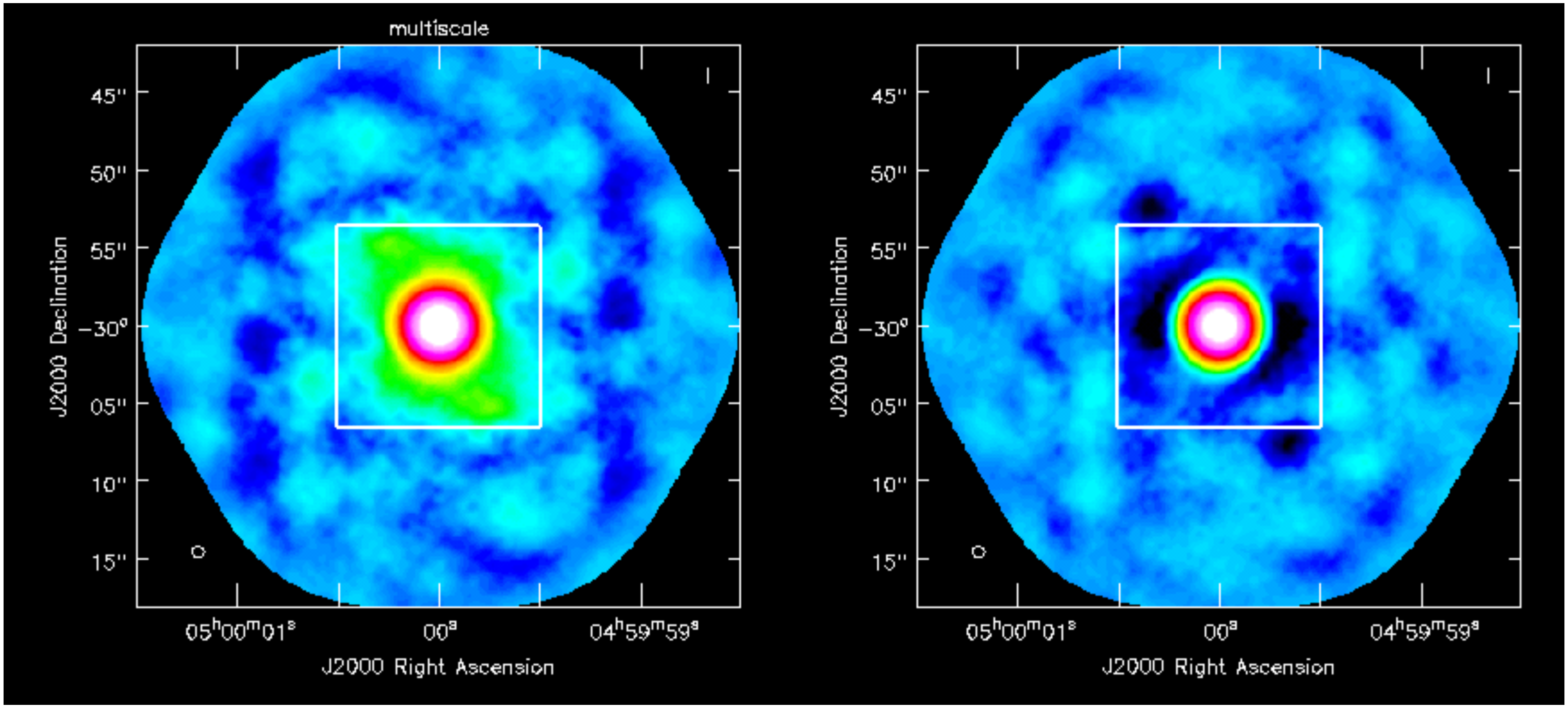
residual (linear scale)



multiscale clean

multiscale

"classic" 1-scale



Miscellaneous

- clean will restart from existing files
  - will first recompute residuals from model
  - be sure this is what you want
- mask image in particular can be reused but be careful of imsize

- total cleaned flux not reported until end
- log messages differ between algorithms (work in progress)

- *don't do ^C while imaging – can do bad things to your MS*

- calready = True required to initialize the MS model column for self-cal

Combining with other data

Single dish or interferometric

If you have only images:

feather

If you have an image and a MS:

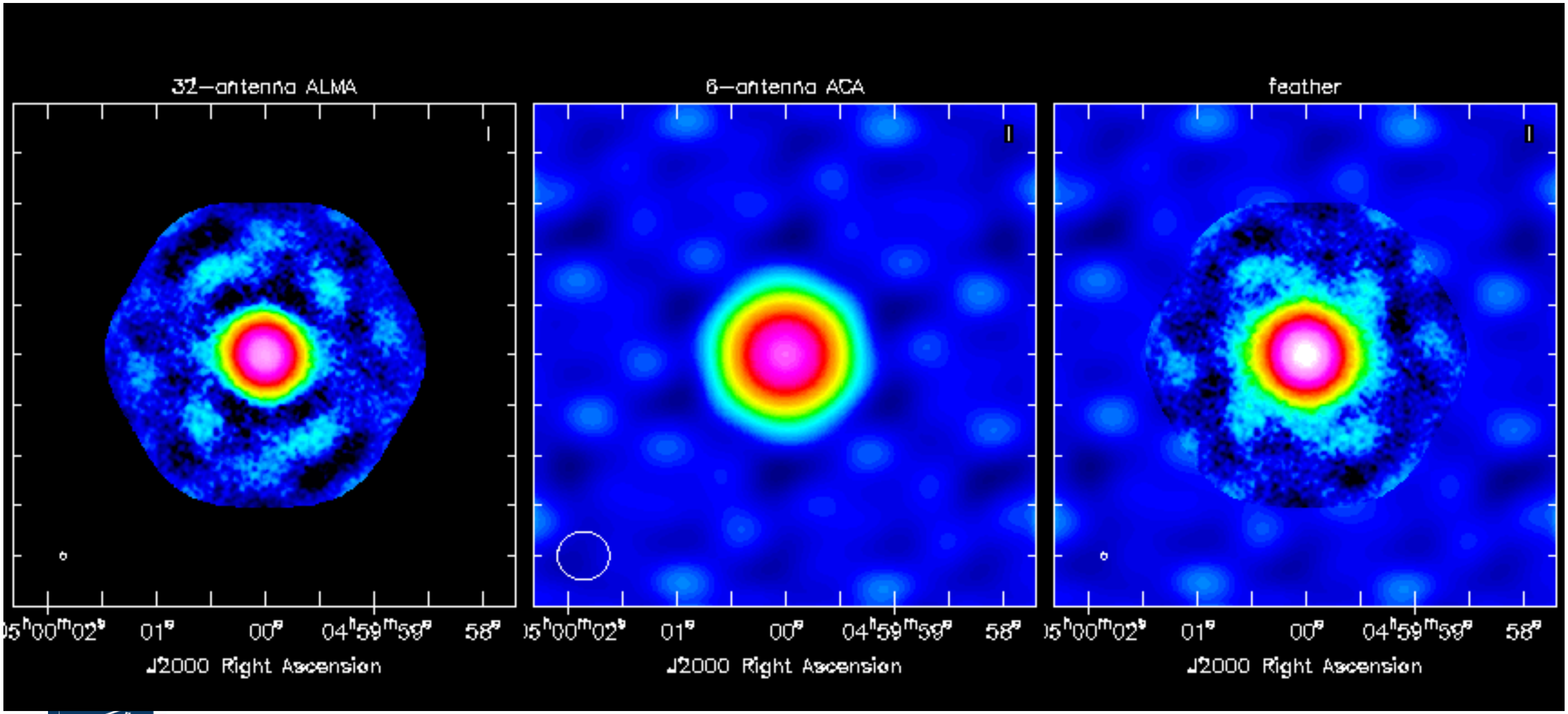
use image as modelimage in clean (or feather)

If you have two+ MS:

concat and deconvolve together (or modelimage or feather)

Combining with other data: feather

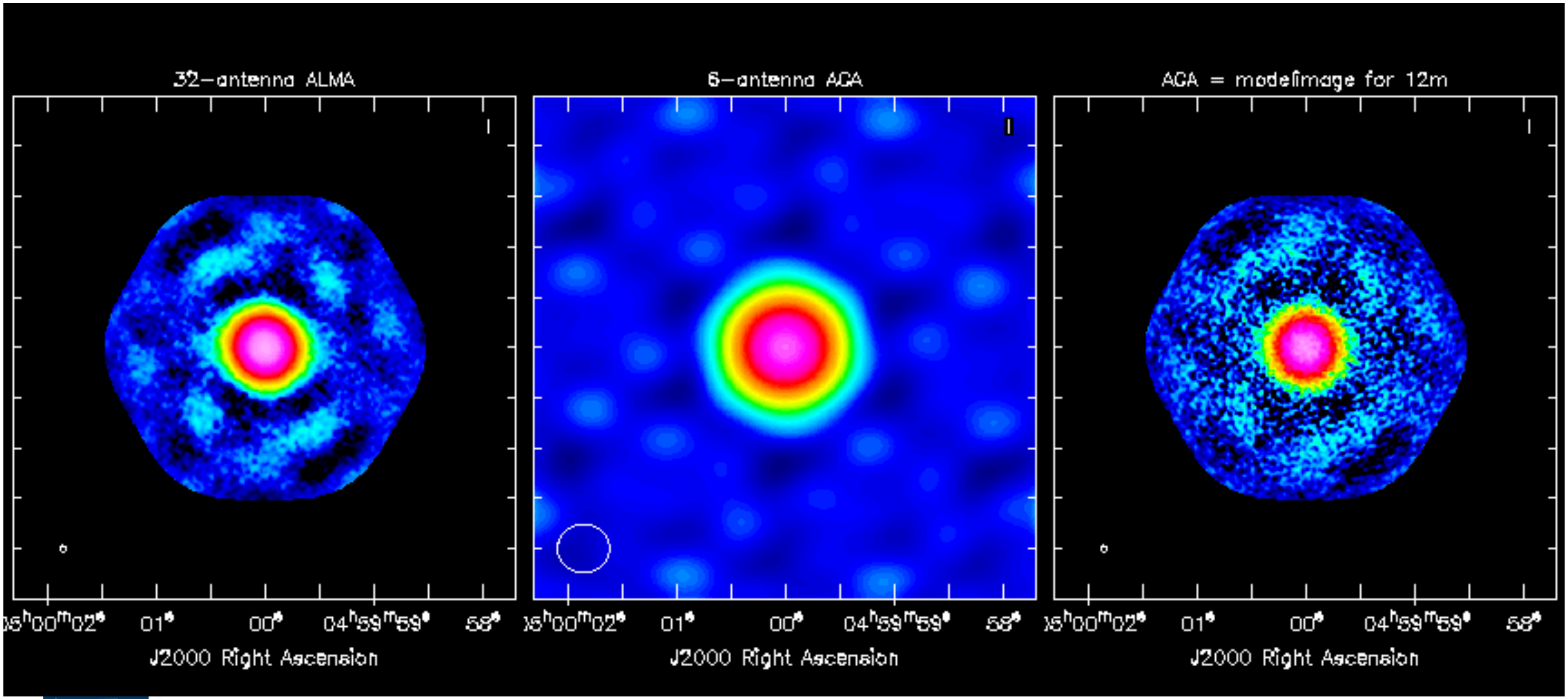
```
#  feather :: Combine two images using their Fourier transforms
imagename           =        ''        #  Name of output feathered image
highres             =        ''        #  Name of high resolution (interferometer) image
lowres              =        ''        #  Name of low resolution (single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



Combining with other data: modelimage

```
--------> inp(clean)
#  clean :: Invert and deconvolve images with selected algorithm
```

```
uvtaper             =      False        #  Apply additional uv tapering of visibilities
modelimage          =         ''        #  Name of model image(s) to initialize cleaning
```



Combining with other data: concat&clean

Theoretically, weights should be scaled by variance, e.g. $(7m/12m)^2(6/32)$

ms tool: ms.getdata[“weight”], ms.putdata

better: use AnalysisUtils

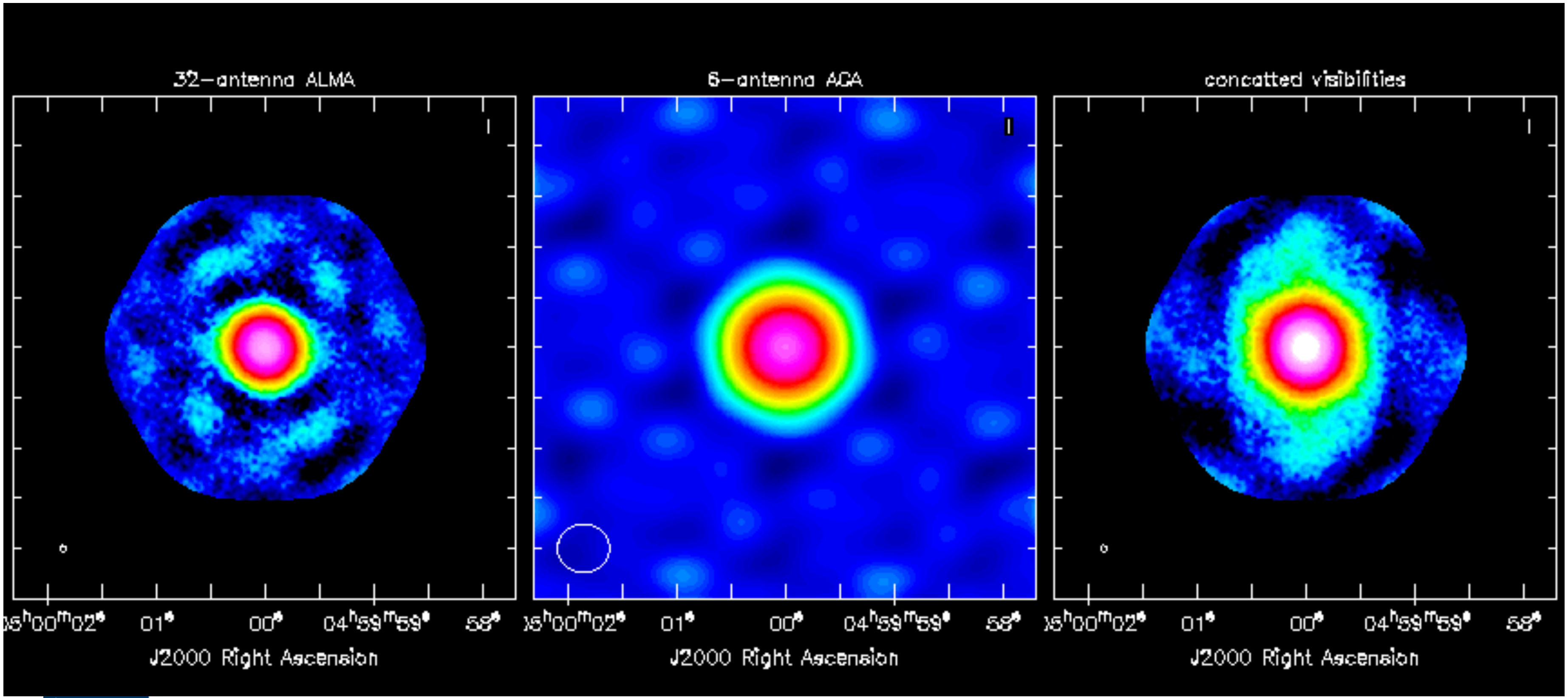


Image Analysis

Analysis and Noise:

- photometry should be done with caution
  - CLEAN does not conserve flux (extrapolates)
  - extended structure missed, attenuated, distorted
  - phase errors (e.g. seeing) can spread signal around

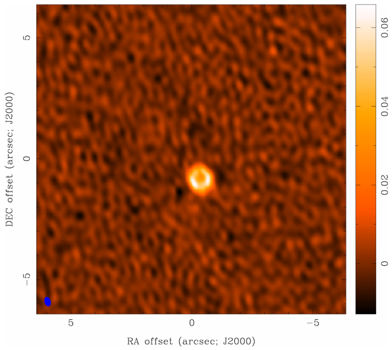
- point source sensitivity: straightforward
  - telescope area, bandwidth, integration time, weighting
  - in image, modify noise by primary beam response
- extended source sensitivity: problematic
  - not quite right to divide noise by $\sqrt{n}$ beams covered by source: smoothing = tapering, omitting data $\rightarrow$ lower limit
  - Interferometers always missing flux at some spatial scale

- be careful with low signal-to-noise images
  - if position known, 3σ OK for point source detection
  - if position unknown, then 5σ required (flux biased by $\sim 1\sigma$)
  - if $< 6\sigma$, cannot measure the source size
  - spectral lines may have unknown position, velocity, width

NRAO

Measures of Image Quality

- "dynamic range"
  - ratio of peak brightness to rms noise in a region devoid of emission (common in astronomy)
  - an easy to calculate ***lower limit*** to the error in brightness in a non-empty region

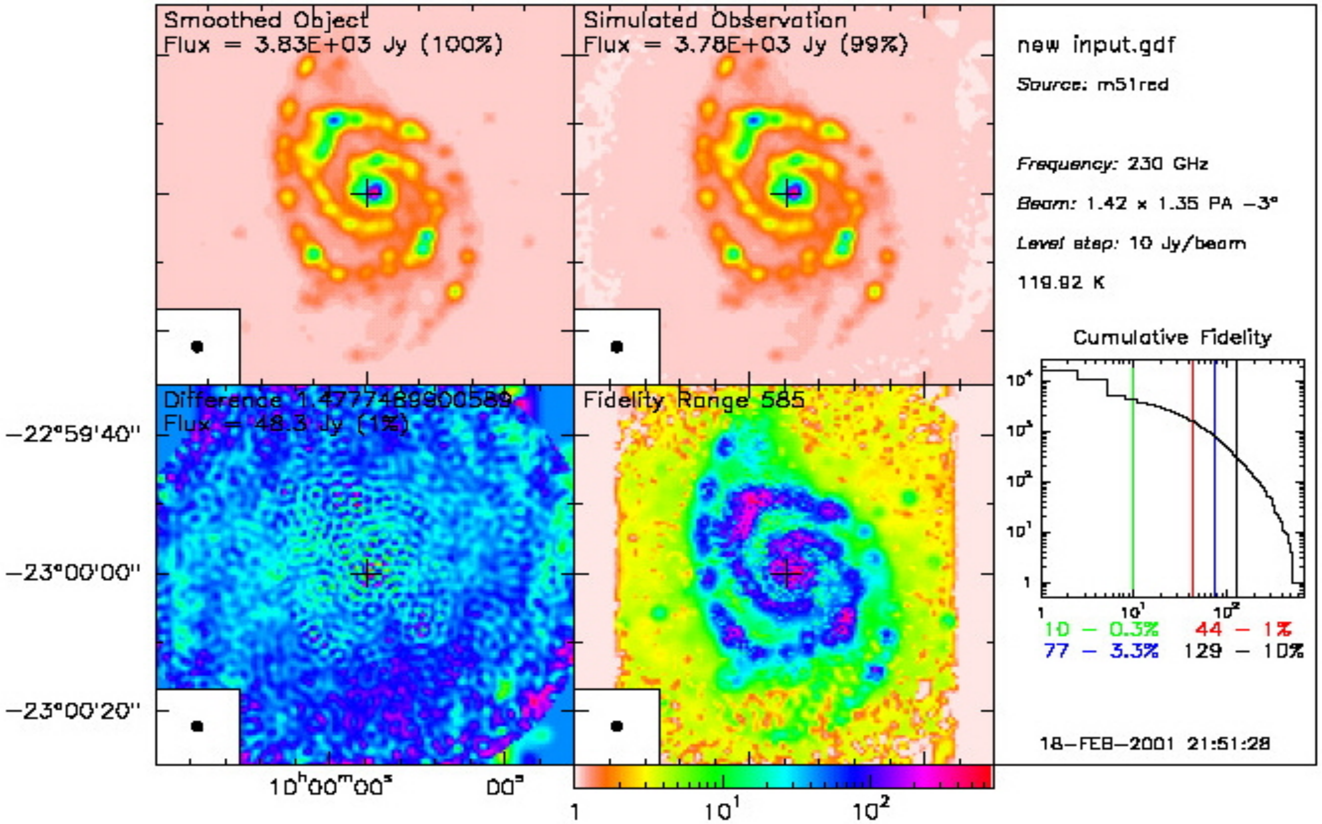


- "fidelity"
  - difference between any produced image and the correct image
  - a convenient measure of how accurately it is possible to make an image that reproduces the brightness distribution on the sky
  - need a priori knowledge of correct image to calculate

  - fidelity image = input model / difference

    = model $\otimes$ beam / abs(model $\otimes$ beam – reconstruction)
  - fidelity is the inverse of the relative error
  - in practice, lowest values of difference need to be truncated

Measures of Image Quality



ALMA Memo #387
Pety et al.

- ALMA Level 1 Science Goal #3
  - ALMA will have: The ability to provide precise images at an angular resolution of 0.1". Here the term precise image means accurately representing the sky brightness at all points where the brightness is greater than 0.1% of the peak image brightness.

Image analysis: casaviewer

Set three mouse buttons to different actions, e.g. zoom, statistics

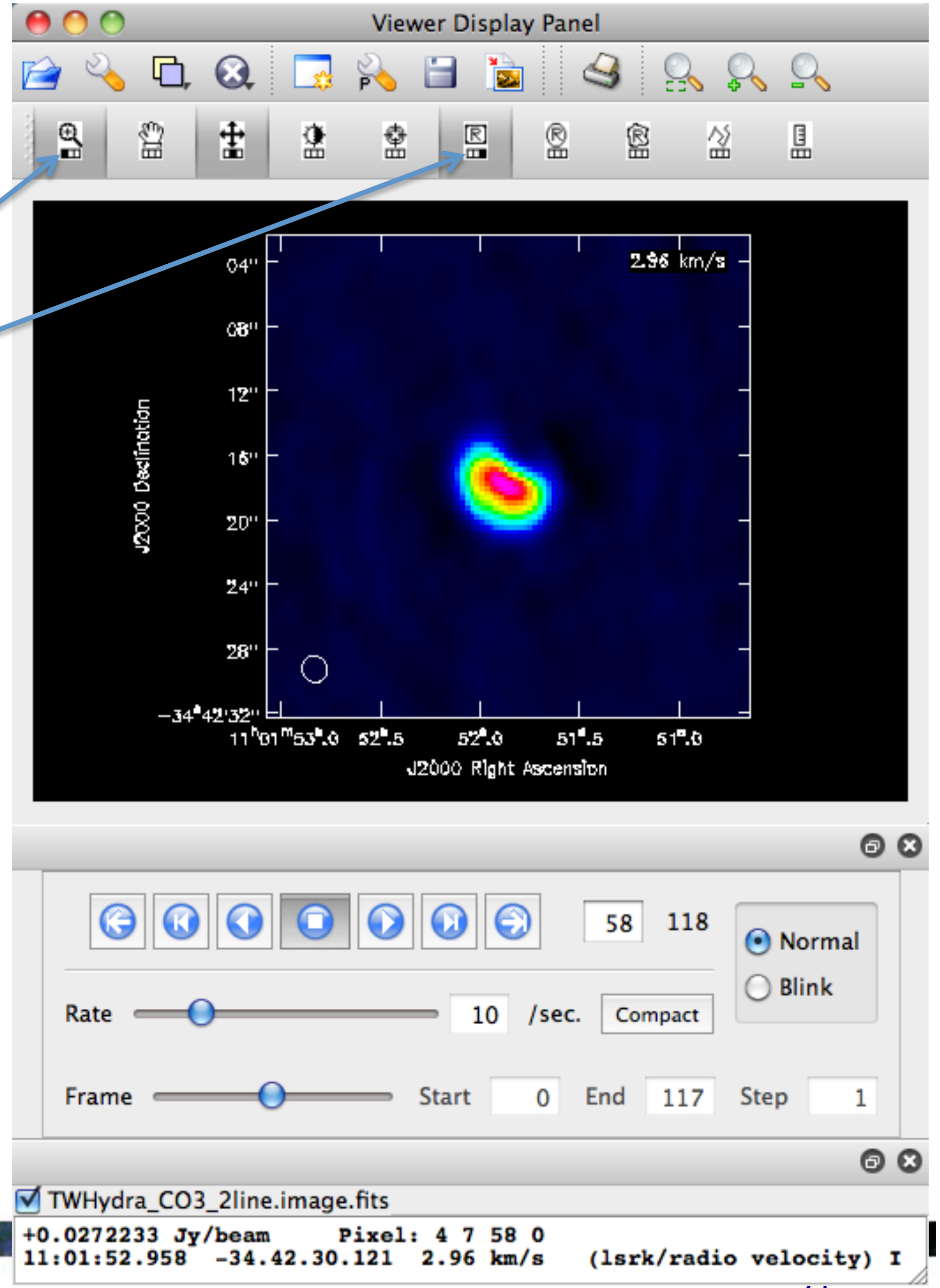
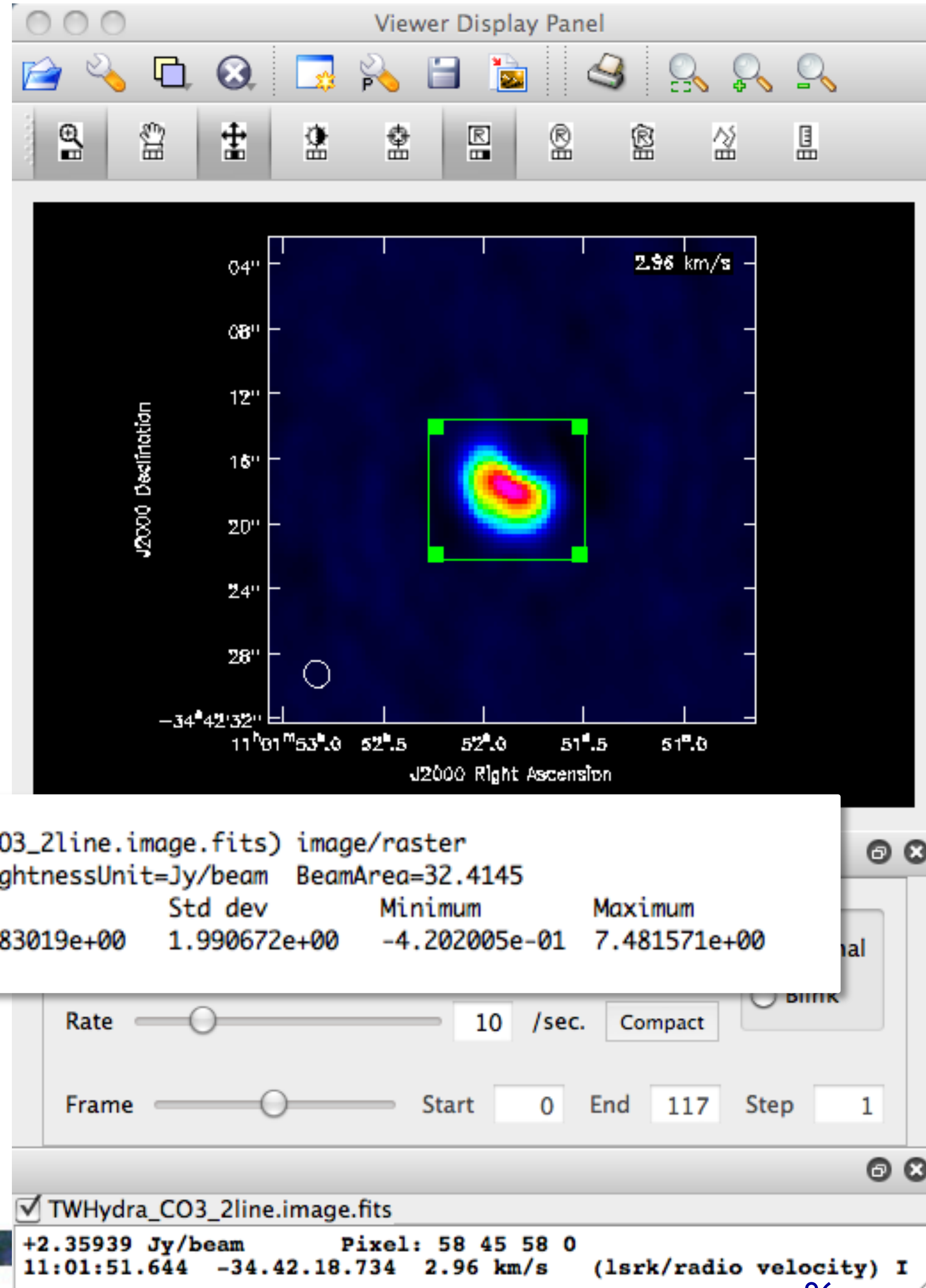


Image analysis: casaviewer

draw with R button, then double R-click

(ESC to remove)



back in your terminal:

```
CASA <5>: (/opt/casa/TWHYA_BAND7_ReferenceImages/TWHydra_CO3_2line.image.fits) image/raster
Frequency=3.457926e+11Hz  Velocity=2.96km/s  Stokes=I  BrightnessUnit=Jy/beam  BeamArea=32.4145
Npts      Sum           Flux (Jy)     Mean          Rms           Std dev       Minimum        Maximum
990       1.298376e+03  4.005542e+01  1.311491e+00  2.383019e+00  1.990672e+00  -4.202005e-01  7.481571e+00
```

Image analysis: casaviewer

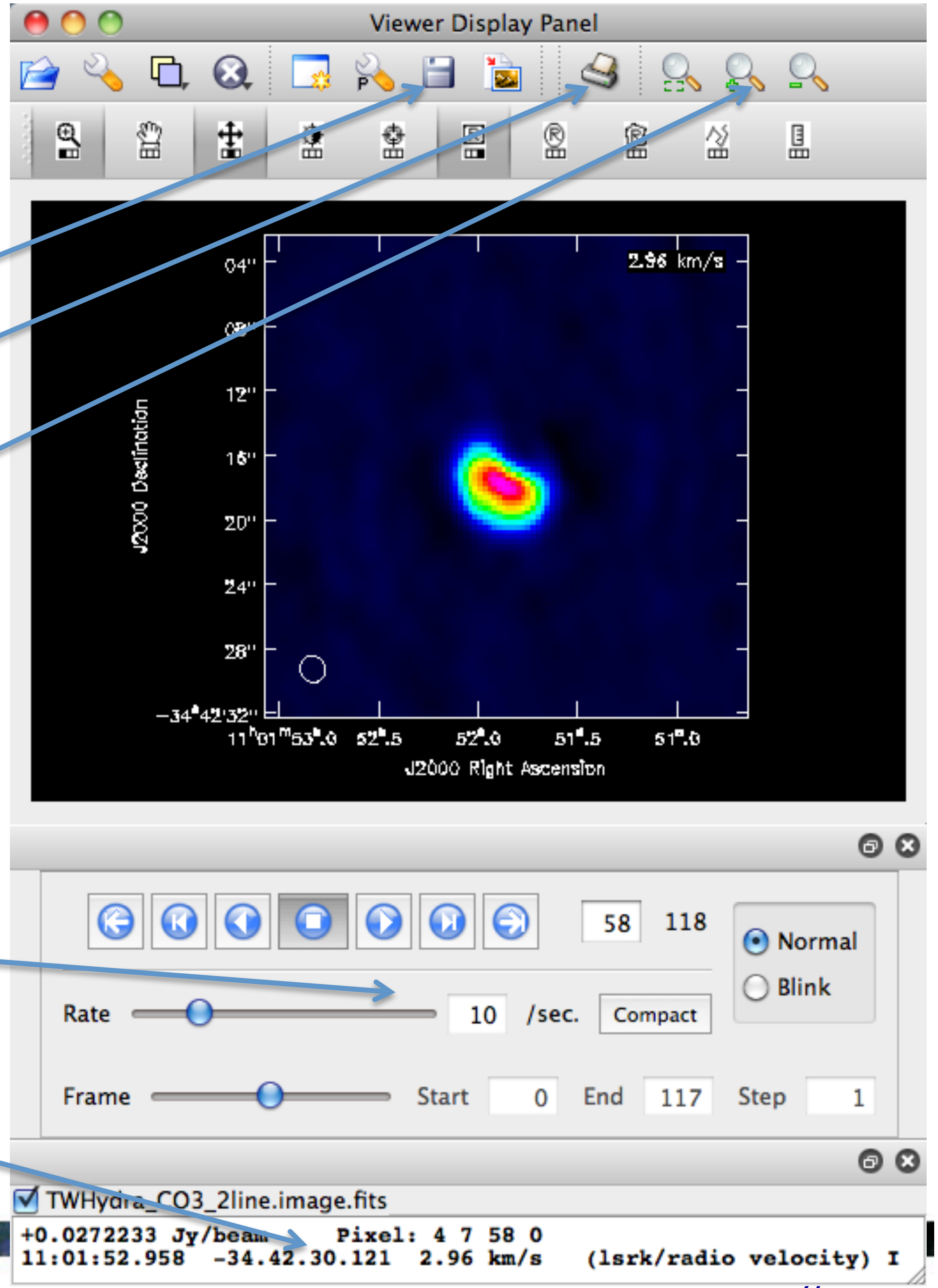
save the current state (images loaded, scale, zoom, etc)

Print or save image to file

zoom to entire image, in/out

Play channels as movie or select one

stats at cursor posn, all loaded images



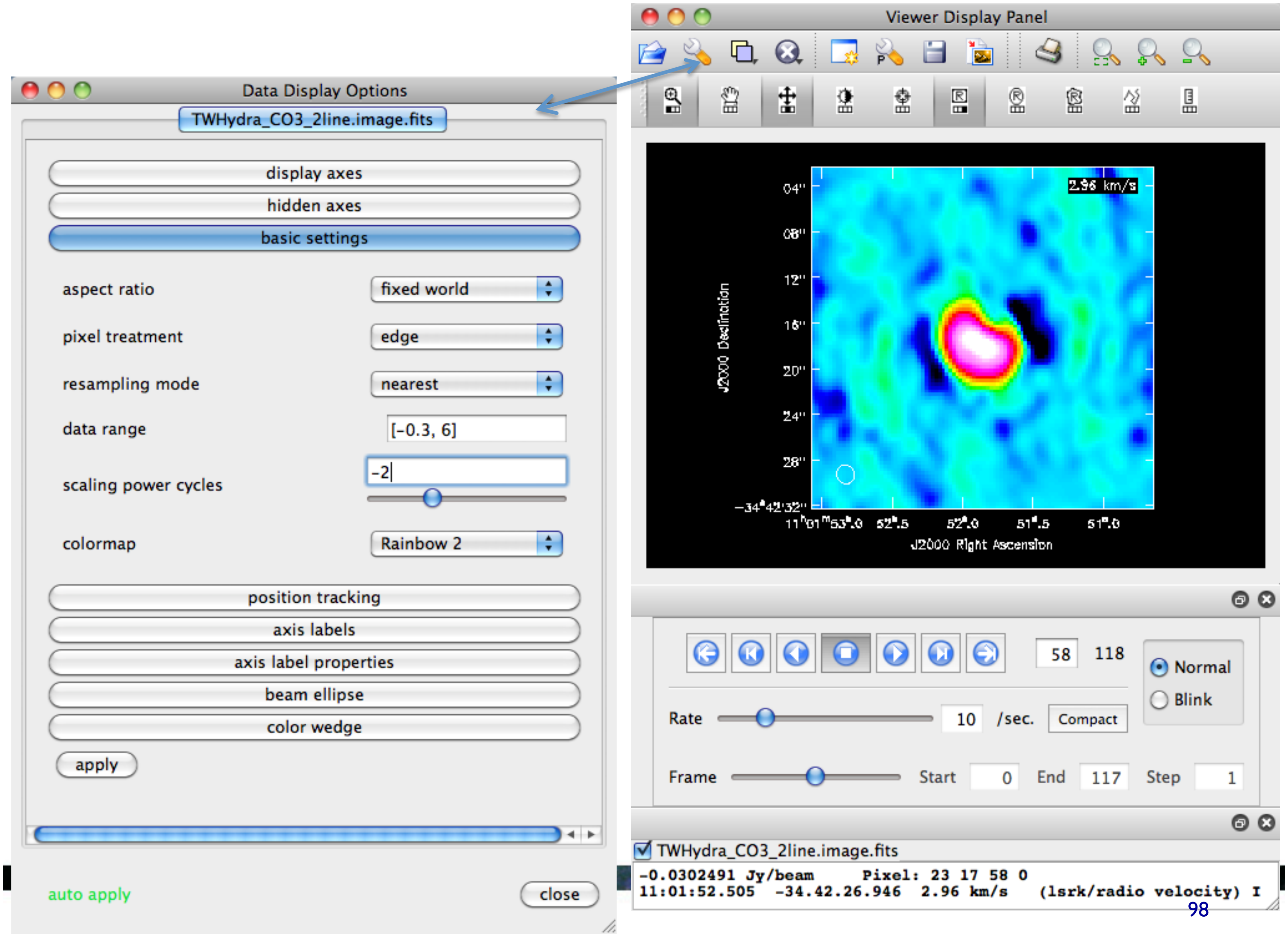
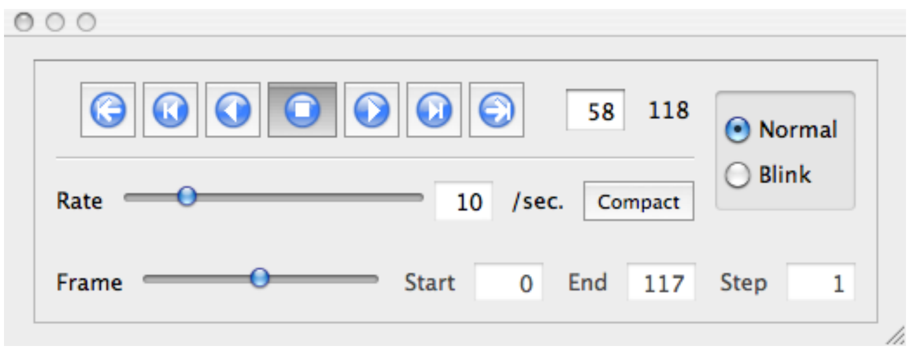

Data Display Options
TWHydra_CO3_2line.image.fits
display axes
hidden axes
basic settings
aspect ratio
fixed world
pixel treatment
edge
resampling mode
nearest
data range
[-0.3, 6]
scaling power cycles
-2
colormap
Rainbow 2
position tracking
axis labels
axis label properties
beam ellipse
color wedge
apply
auto apply
close
Viewer Display Panel
2.96 km/s
J2000 Declination
J2000 Right Ascension
58
118
Normal
Blink
Rate
10
/sec.
Compact
Frame
Start
0
End
117
Step
1
TWHydra_CO3_2line.image.fits
-0.0302491 Jy/beam Pixel: 23 17 58 0
11:01:52.505 -34.42.26.946 2.96 km/s (lsrk/radio velocity) I

Image analysis: casaviewer

- subpanels can be removed (and redocked)



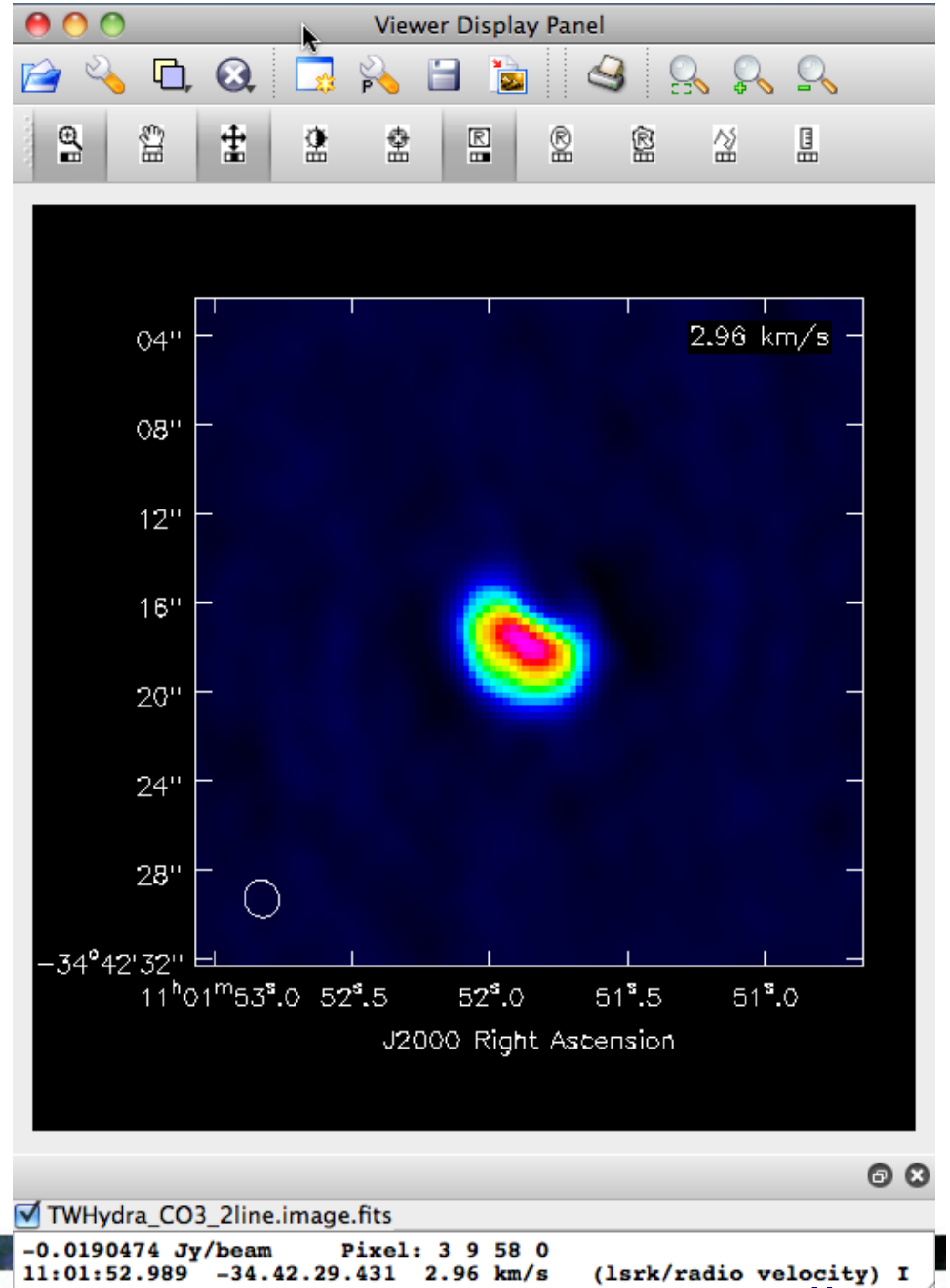
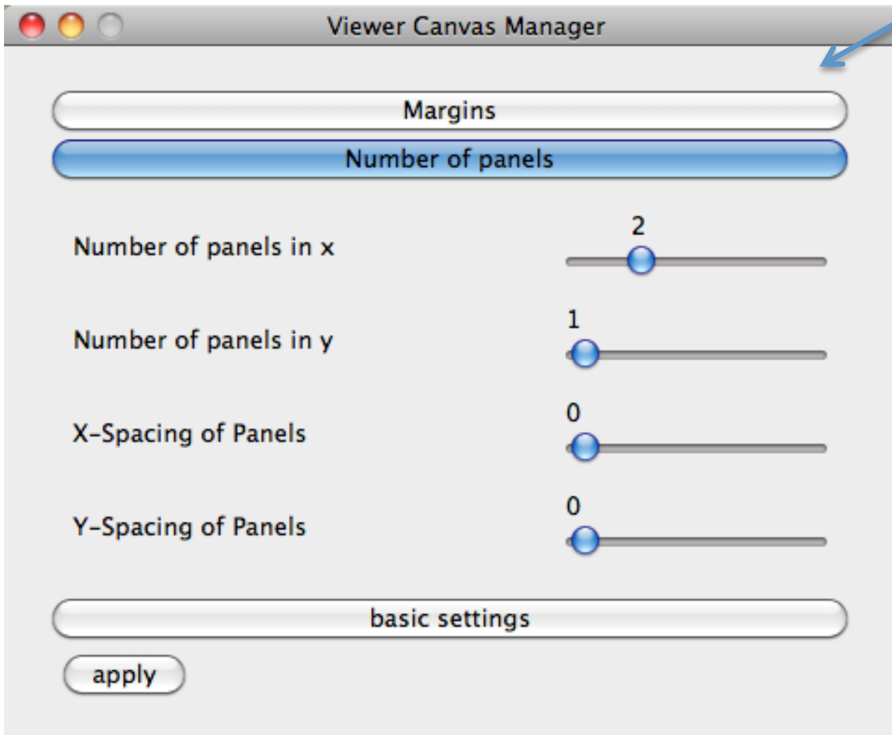


Image analysis: casaviewer



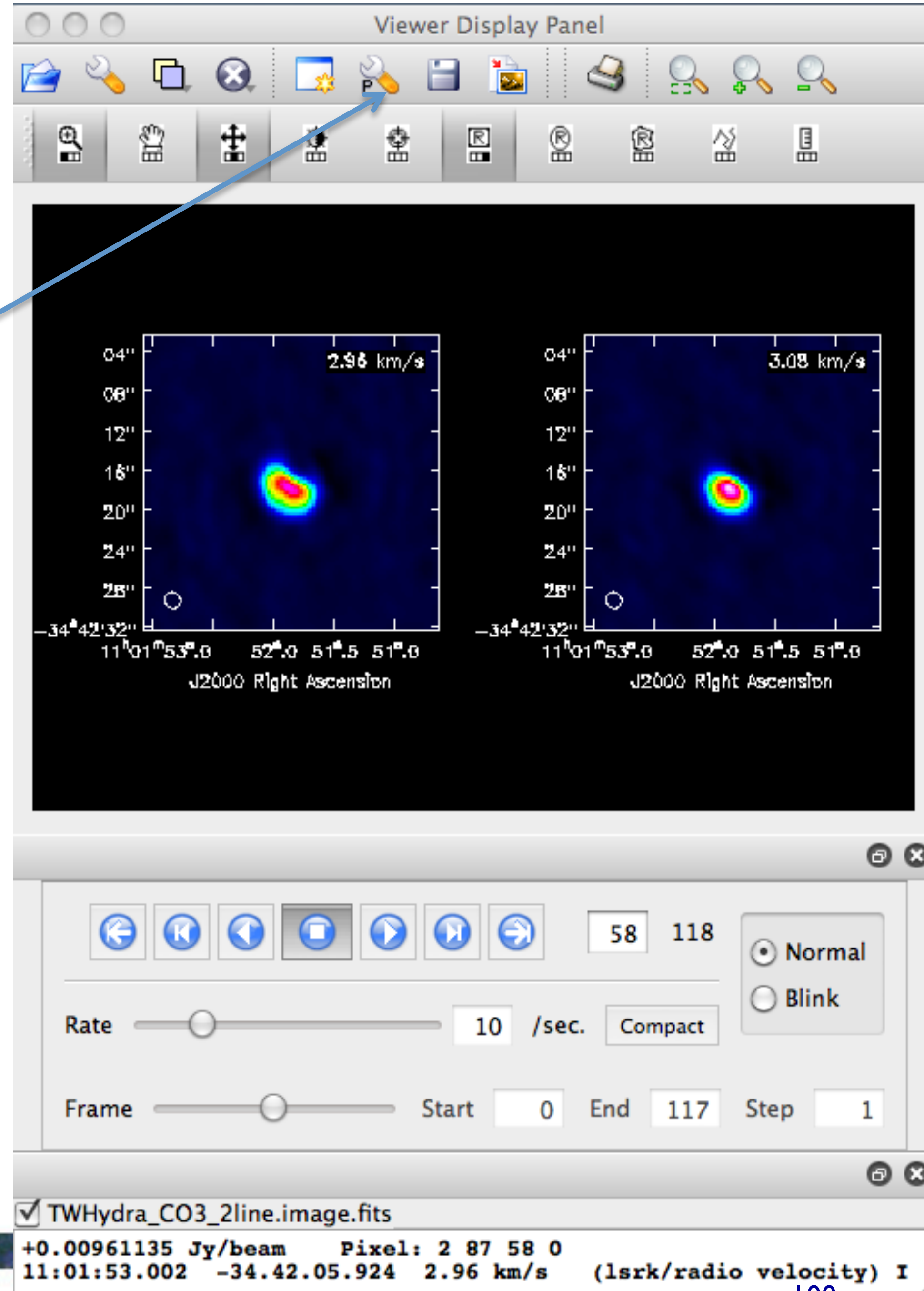


Image analysis: casaviewer

spectral profile tool

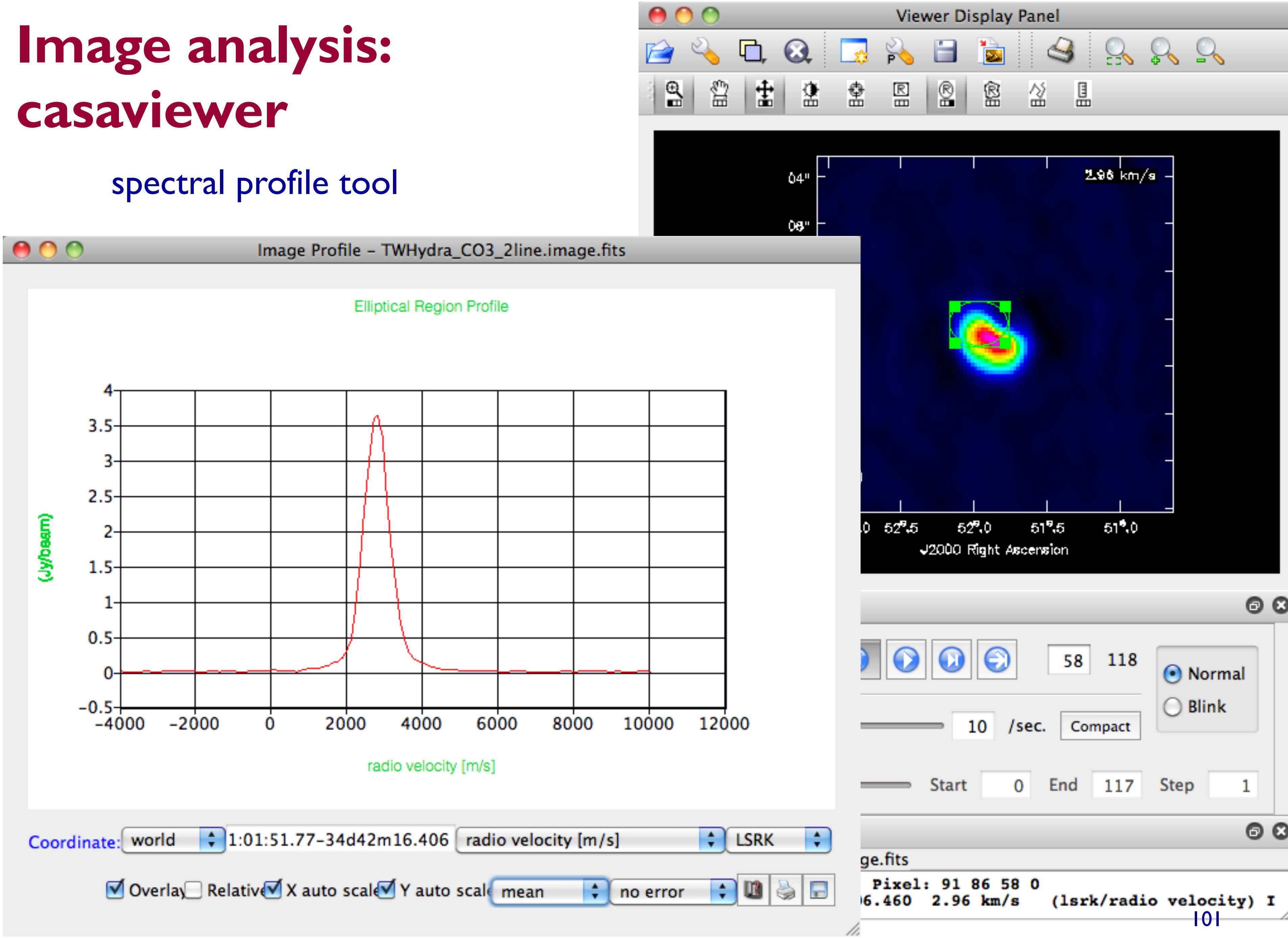


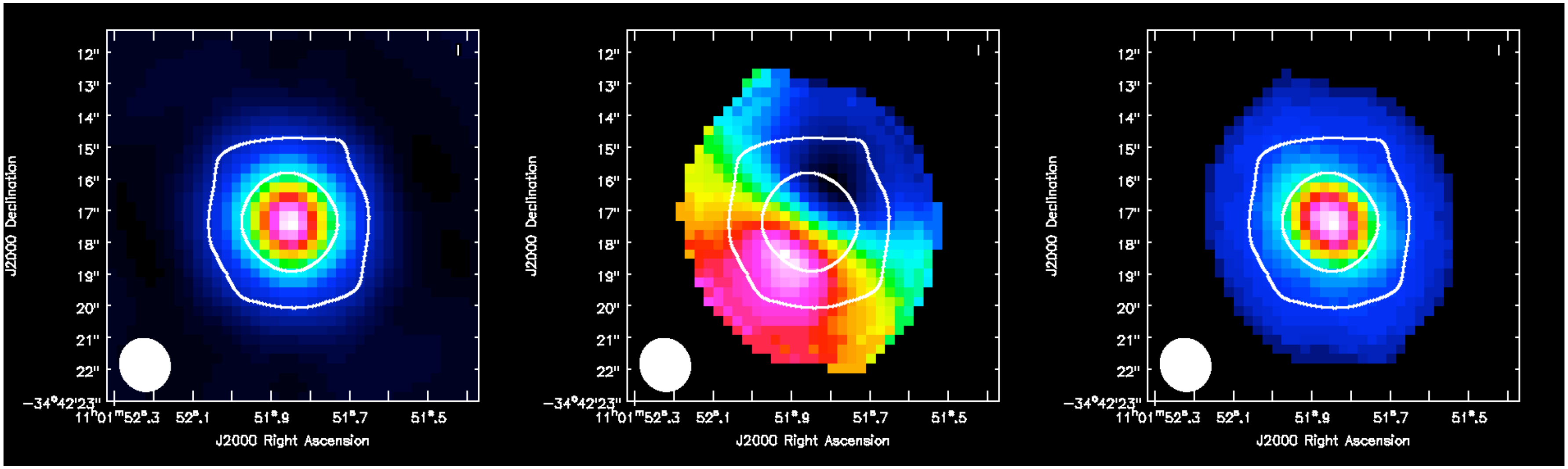
Image analysis: immoments

```
#  immoments :: Compute moments from an image
imagename      =         ''        #  Name of the input image
moments        =        [0]        #  List of moments you would like to compute
axis           = 'spectral'        #  The momement axis: ra, dec, lat, long, spectral, or stokes
region         =         ''        #  Image Region.  Use viewer
box            =         ''        #  Select one or more box regions
chans          =         ''        #  Select the channel(spectral) range
stokes         =         ''        #  Stokes params to image (I,IV,IQU,IQUV)
mask           =         ''        #  mask used for selecting the area of the image to calculate the moments on
includepix     =         -1        #  Range of pixel values to include
excludepix     =         -1        #  Range of pixel values to exclude
outfile        =         ''        #  Output image file name (or root for multiple moments)
async          =      False        #  If true the taskname must be started using immoments(...)
```

Set relatively high threshold for higher moments e.g. 6σ measured in the line-peak channel with the viewer

```
moments=-1  - mean value of the spectrum
moments=0   - integrated value of the spectrum
moments=1   - intensity weighted coordinate;traditionally used to get
              'velocity fields'
moments=2   - intensity weighted dispersion of the coordinate; traditionally
              used to get "velocity dispersion"
moments=3   - median of I
moments=4   - median coordinate
moments=5   - standard deviation about the mean of the spectrum
moments=6   - root mean square of the spectrum
moments=7   - absolute mean deviation of the spectrum
moments=8   - maximum value of the spectrum
moments=9   - coordinate of the maximum value of the spectrum
moments=10  - minimum value of the spectrum
moments=11  - coordinate of the minimum value of the spectrum
```

Image analysis: immoments




```
immoments(imagename='TWHydra_CO3_2line.image',moments=[0],
          outfile='TWHydra_CO3_2line.image.mom0',
          chans='40~76')
```

```
immoments(imagename='TWHydra_CO3_2line.image',moments=[1,2],
          outfile='TWHydra_CO3_2line.image.mom',
          chans='40~76',includepix=[0.3,100])
```

resources

Interferometry

- Virtual Radio Interferometer http://www.narrabri.atnf.csiro.au/astronomy/vri.html
- NRAO synthesis school lectures http://www.aoc.nrao.edu/events/synthesis/2010/
- IRAM lectures http://www.iram-institute.org/EN/content-page-204-7-67-202-204-0.html

CASA

- casa.nrao.edu
- casaguides.nrao.edu