Imaging ALMA data

Preparation + hands-on demo

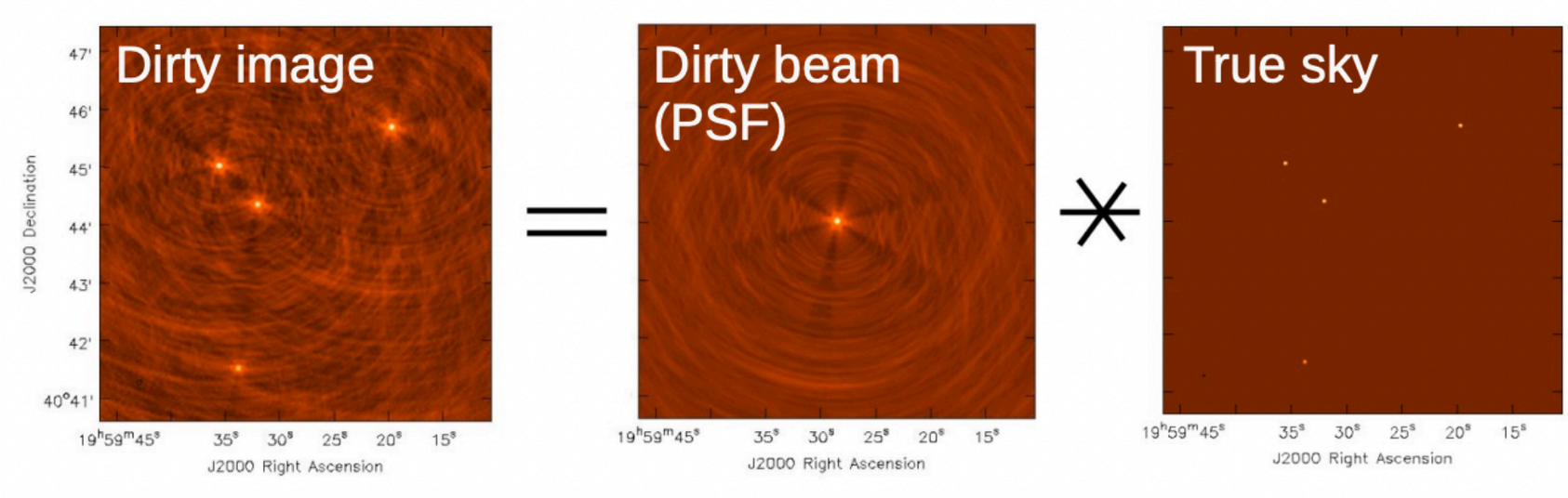
Dan Walker

EUROPEAN ARC
ALMA Regional Centre || UK

MANCHESTER 1824

How do we actually image ALMA data?

- With an interferometer, we observe an interference pattern that is expressed by the complex visibility $V(u,v)$
- Inverse Fourier transform of $V(u,v)$ → image in (x,y)
  - Image has complicated artefacts due to incomplete sampling of uv-plane. This is the so-called '**dirty image**' — a convolution of the sky brightness and the '**dirty beam**' (point spread function).
- Solution is to deconvolve dirty beam from dirty image to recover the true sky brightness



Deconvolution

- Dirty image is not ideal for science due to image artefacts
- Deconvolution aims to reconstruct the true sky brightness, but …
  - Missing information due to incomplete uv-coverage
  - Data is corrupted by noise
  - There is no unique solution

→ Aim is to find a good model of the sky brightness

Most widely used deconvolution method is the **CLEAN** algorithm (Hogbom 1974)

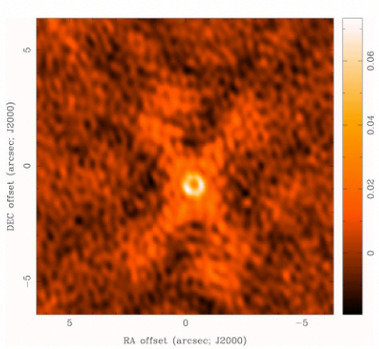
Basic CLEAN algorithm

Initialise a residual map (dirty map)

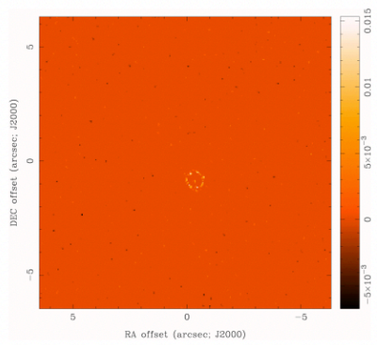
- Identify strongest feature in residual map as a point source
- Add point source to the clean model
- Convolve the point source with dirty beam and subtract from residual map
- If stopping criteria not reached, do next iteration

Convolve clean model with the ‘clean beam’ (usually a Gaussian estimated from the dirty beam) and add residual map to make the final image

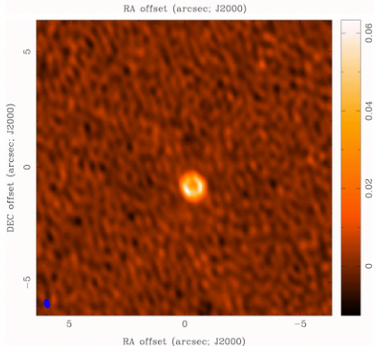
Dirty image



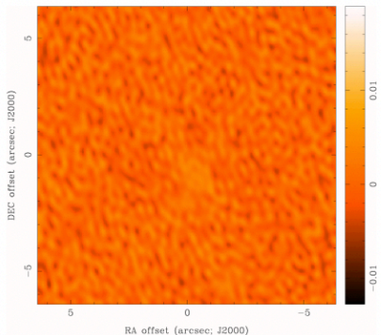
Clean model



Cleaned image



Residual



Target: PJ113921.7

- Dusty star-forming galaxy
- Redshift (z) = 2.858
- Gravitationally-lensed galaxy

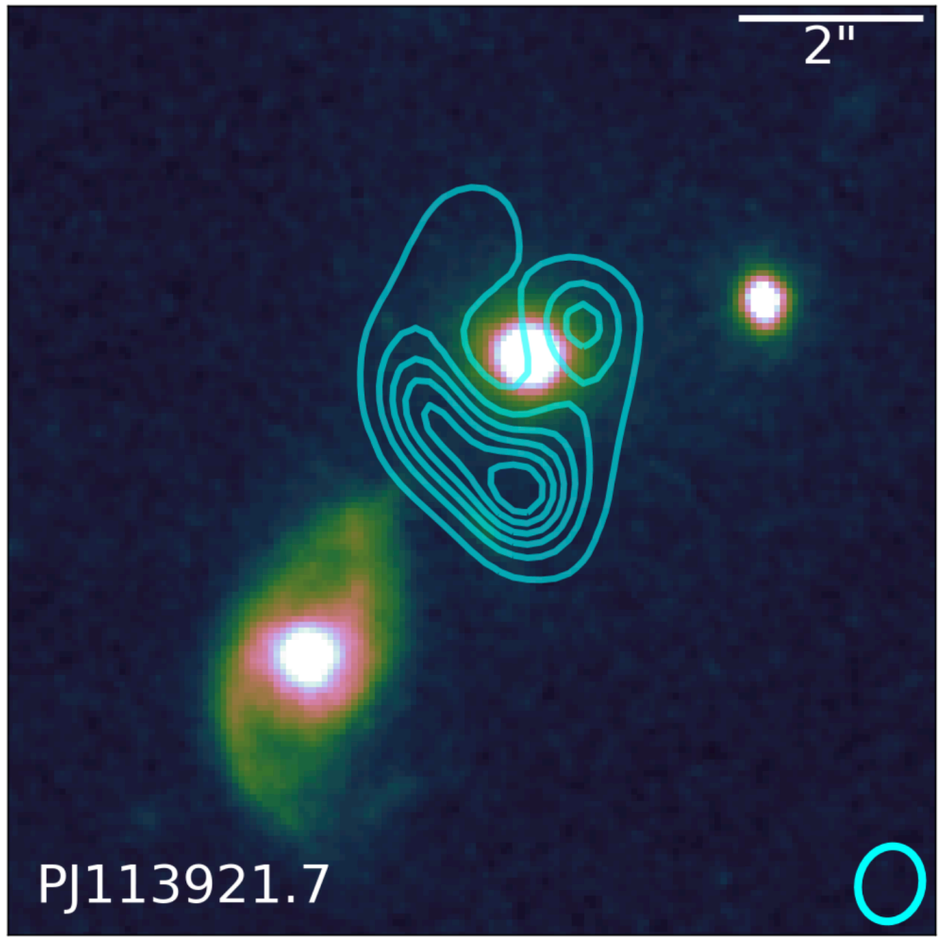


Image: ALMA 1 mm dust continuum

Contours: Hubble 1.6 μm

[Source: ALMA proposal 2021.1.00499.S]

PJ113921.7 ALMA data

- ALMA project ID: 2021.1.00499.S
- Band 3 (~ 87 - 102.5 GHz)
- 3x Continuum SPWs & 1x CO (3-2) SPW
- Two different ALMA 12m observations:
  - TM1: longer baseline / higher angular resolution
  - TM2: shorter baseline / lower angular resolution
- We will use the **TM2** data only (smaller -> faster processing)

PJ113921.7 ALMA data

uid___A002_Xf396d6_X45bb.ms

- You should have this calibrated MS if you ran `scriptForPI.py`
- Full calibrated MS can be downloaded here (~ 23 GB)
- Science target only MS is available here (~ 6.2 GB)

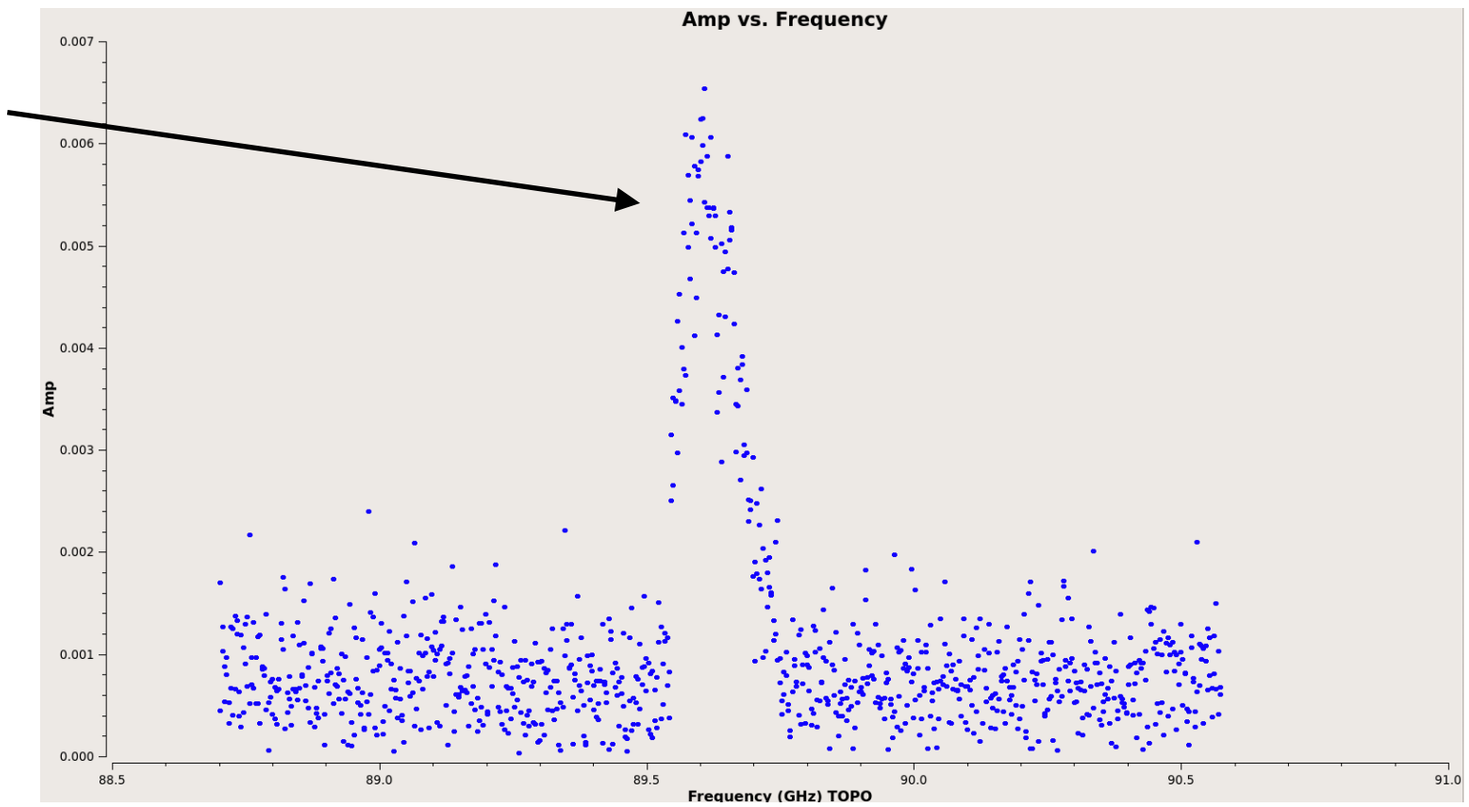
imaging_script.py

- This is the script that we will walk through in this example to clean the continuum and CO data
- http://almanas.jb.man.ac.uk/alma/Web/Meetings/2023/UKHybridWorkshop/imaging_script.py

Continuum imaging

- Before imaging the continuum, we need to identify any molecular line emission — this will contaminate the continuum if we don't exclude it
  - For this example, we will use the line frequency ranges identified by the ALMA pipeline* (`contchans` parameter in the imaging script)
  - Another method is to look at each spectral window using the CASA task `plotms`, and manually identifying the continuum ranges:

- This is the CO line in spectral window 25 of our example dataset, which should not be included when generating the continuum image



Continuum imaging

To image the data, we will use the CASA task `tclean`. We'll start by making a 'dirty image' (0 clean iterations) — this will give us a first look at the data and allow us to refine our choice of parameters. Let's take a look at some of the parameters:

```
tclean( vis             = visfile,

        imagename       = 'PJ113921.7.cont.dirty',

        field           = 'PJ113921.7',

        spw             = contchans,

        specmode        = 'cont',

        imsize          = [1250, 1250],

        cell            = '0.09arcsec',

        deconvolver     = 'hogbom',

        niter           = 0,

        weighting       = 'briggs',

        robust          = 0.5,

        interactive     = False)
```

Name of input measurement set

Prefix of output image files

Field name to be imaged

Channel ranges to be used to generate continuum

Spectral mode ('cont' for continuum)

Image size (2*Field of view / pixel size)

Pixel size (generally ~ θ/5, θ = angular resolution)

Deconvolution algorithm to be used

Number of clean iterations

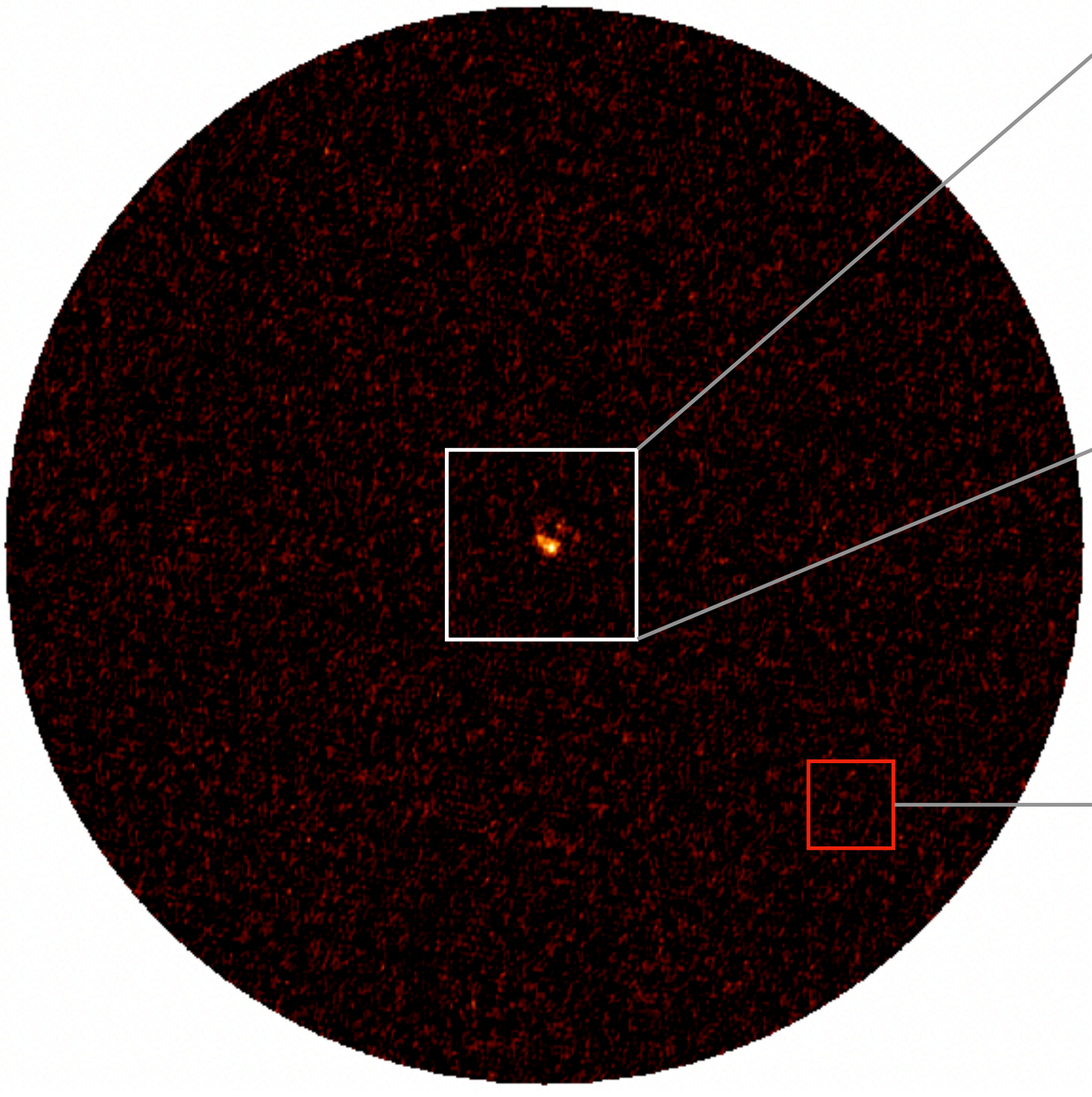
Weighting scheme to be used

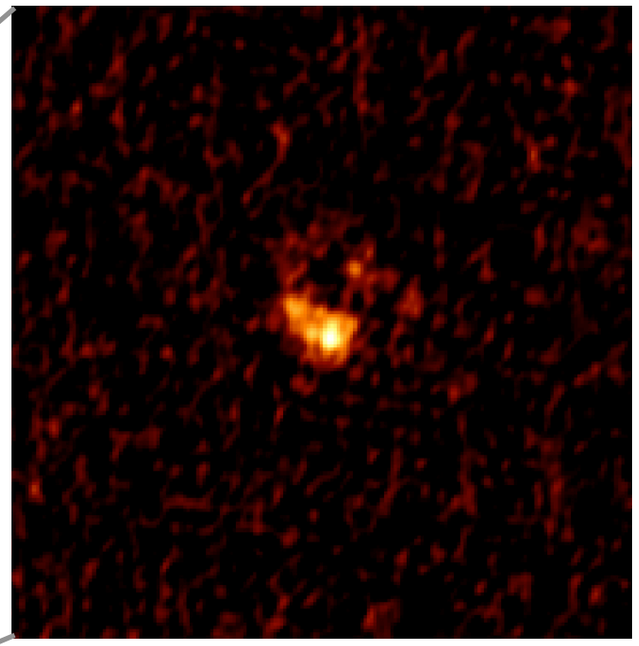
Robust parameter for Briggs weighting (robust = -2 gives uniform weighting. robust = 2 gives natural weighting)

Option to clean using interactive GUI

Continuum imaging

Dirty image





Field is mostly blank

Lensed galaxy clearly detected in centre of field

RMS ~ 1.3e-5 Jy

Continuum imaging

- Based on the RMS in our dirty image, we can specify a sensible cleaning threshold, typically N x RMS, where N is typically ~ 1-5

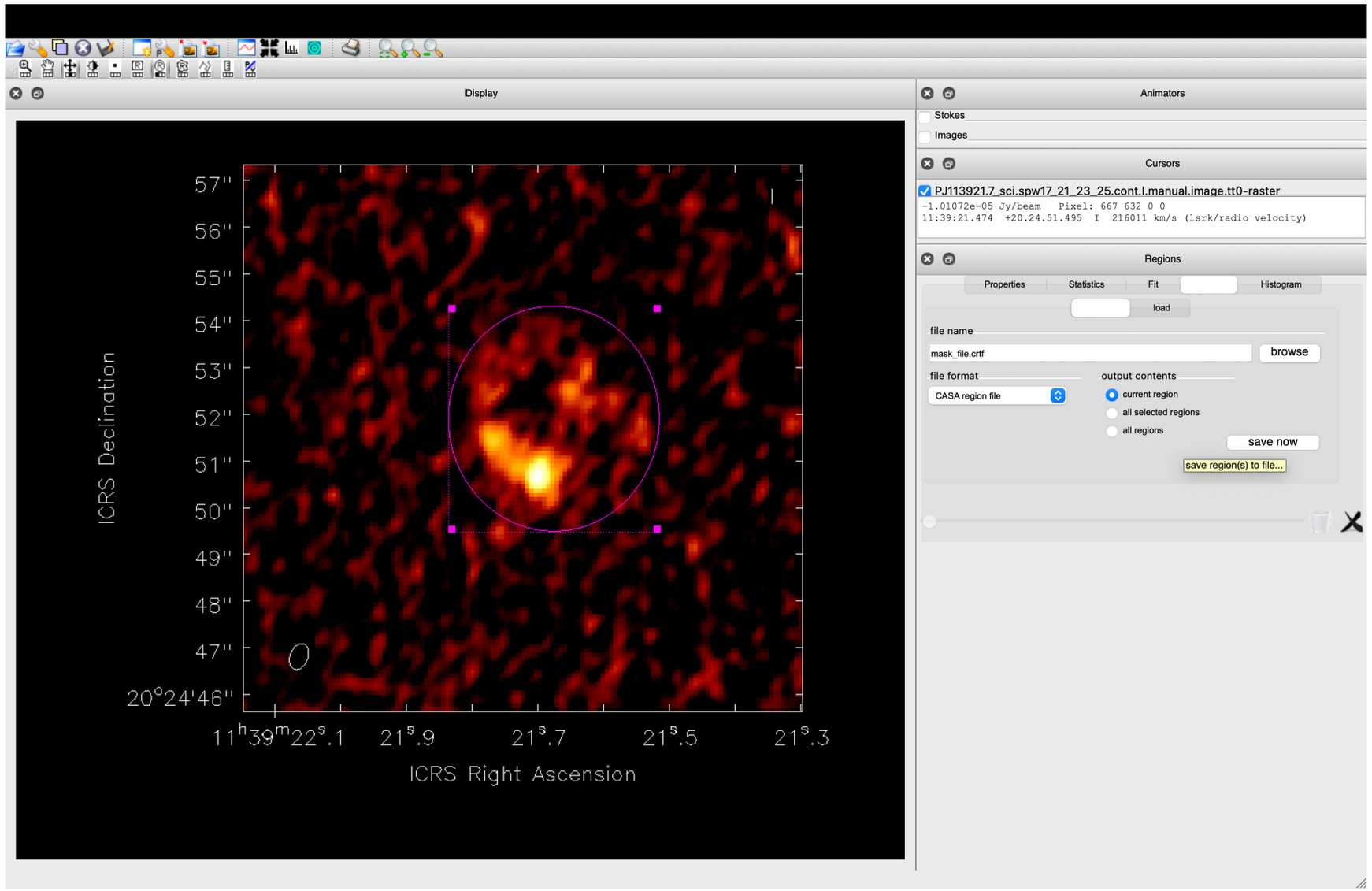
- Need to set number of clean iterations. If the cleaning threshold is sensible, this can be arbitrarily high, as the cleaning should stop once the threshold has been reached
    - A poor choice of cleaning parameters may lead to divergence and general weirdness!

- For this demo, the image size has been decreased to the central 150x150 pixels — this excludes the blank areas and speeds things up

- Finally, a quick note on (auto-)masking, weighting, and primary-beam correction …

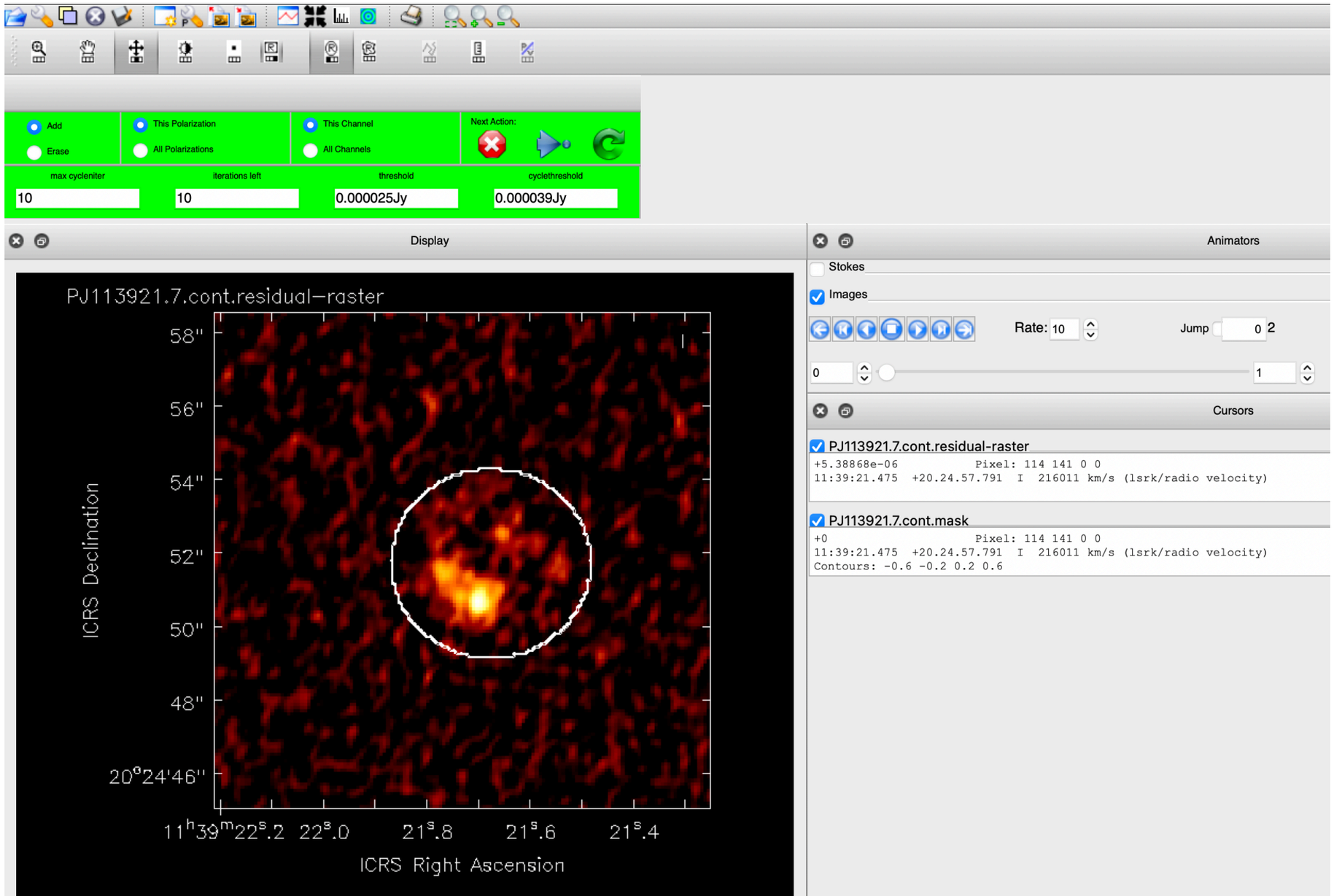
Continuum imaging (masking)

- We can use a cleaning mask to tell the algorithm where there is real emission to be cleaned. This can be done by:
  - Providing a pre-made mask as a cleaning parameter



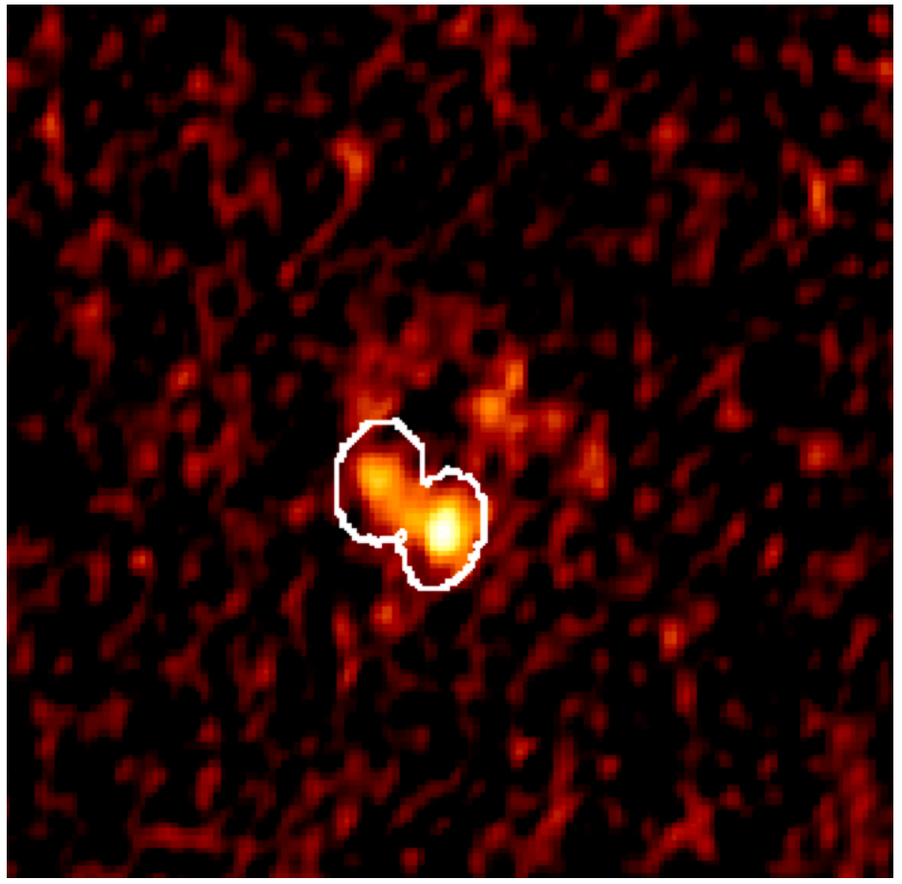
Continuum imaging (masking)

- We can use a cleaning mask to tell the algorithm where there is real emission to be cleaned. This can be done by:
  - Using the interactive cleaning GUI to manually draw a mask



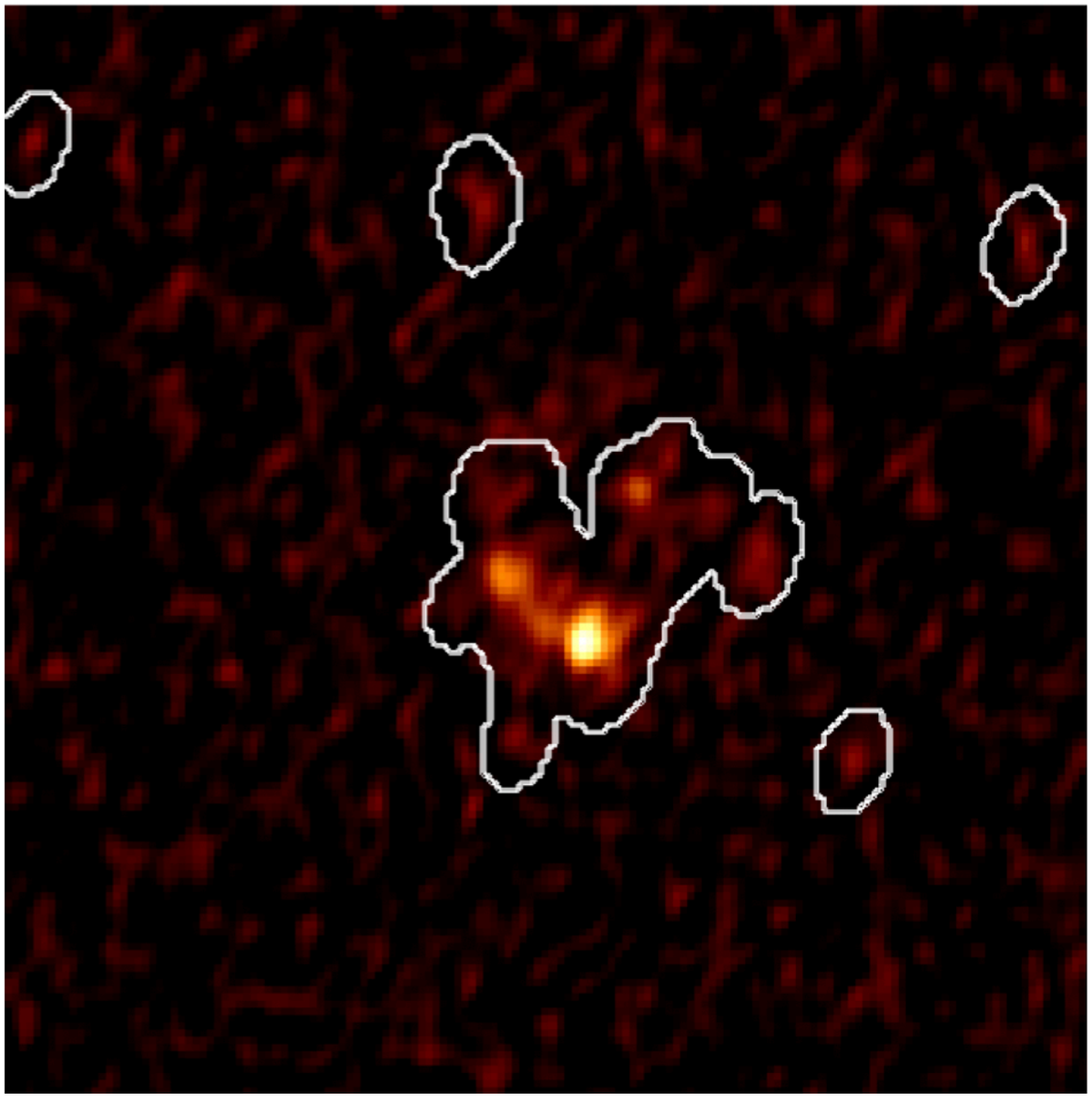
Continuum imaging (masking)

- We can use a cleaning mask to tell the algorithm where there is real emission to be cleaned. This can be done by:
  - Using the built-in auto-masking functionality
  - https://casaguides.nrao.edu/index.php/Automasking_Guide
- Much more convenient, and can do a very good job
- Requires careful choice of parameters — the default parameters *typically* do a reasonable job, but can often be improved

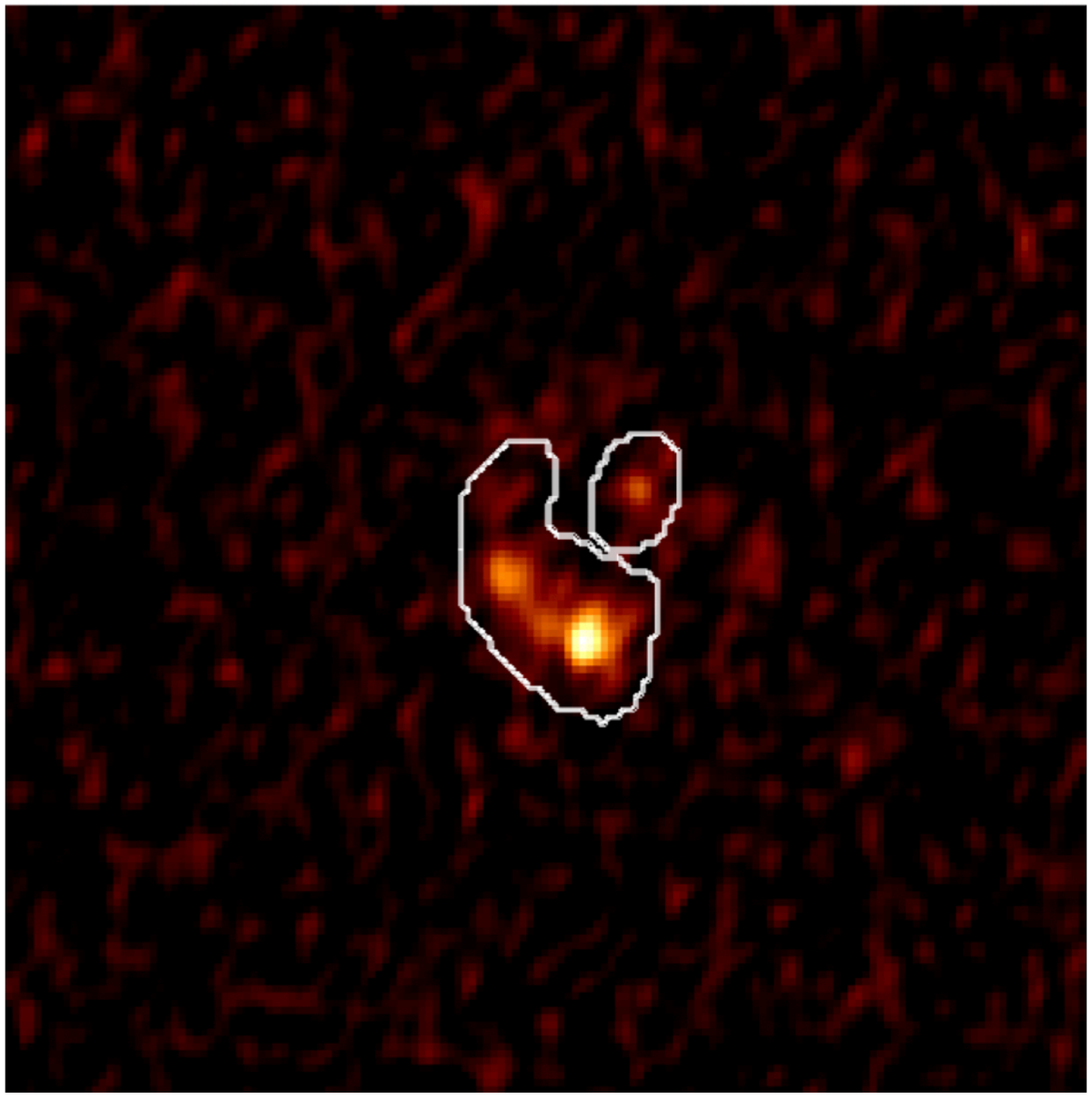


Automasking

Need to be more strict with masking thresholds

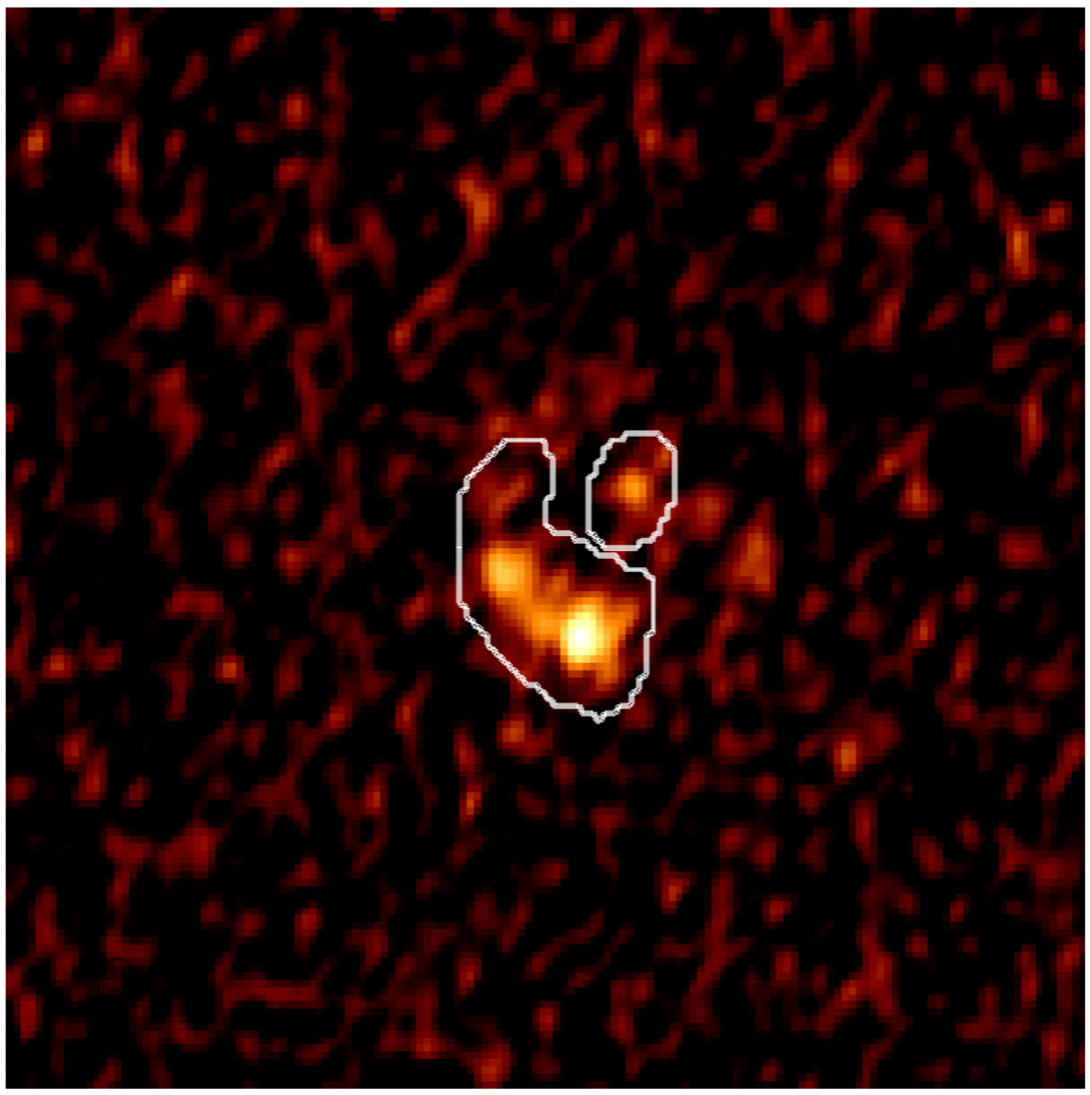


Better …

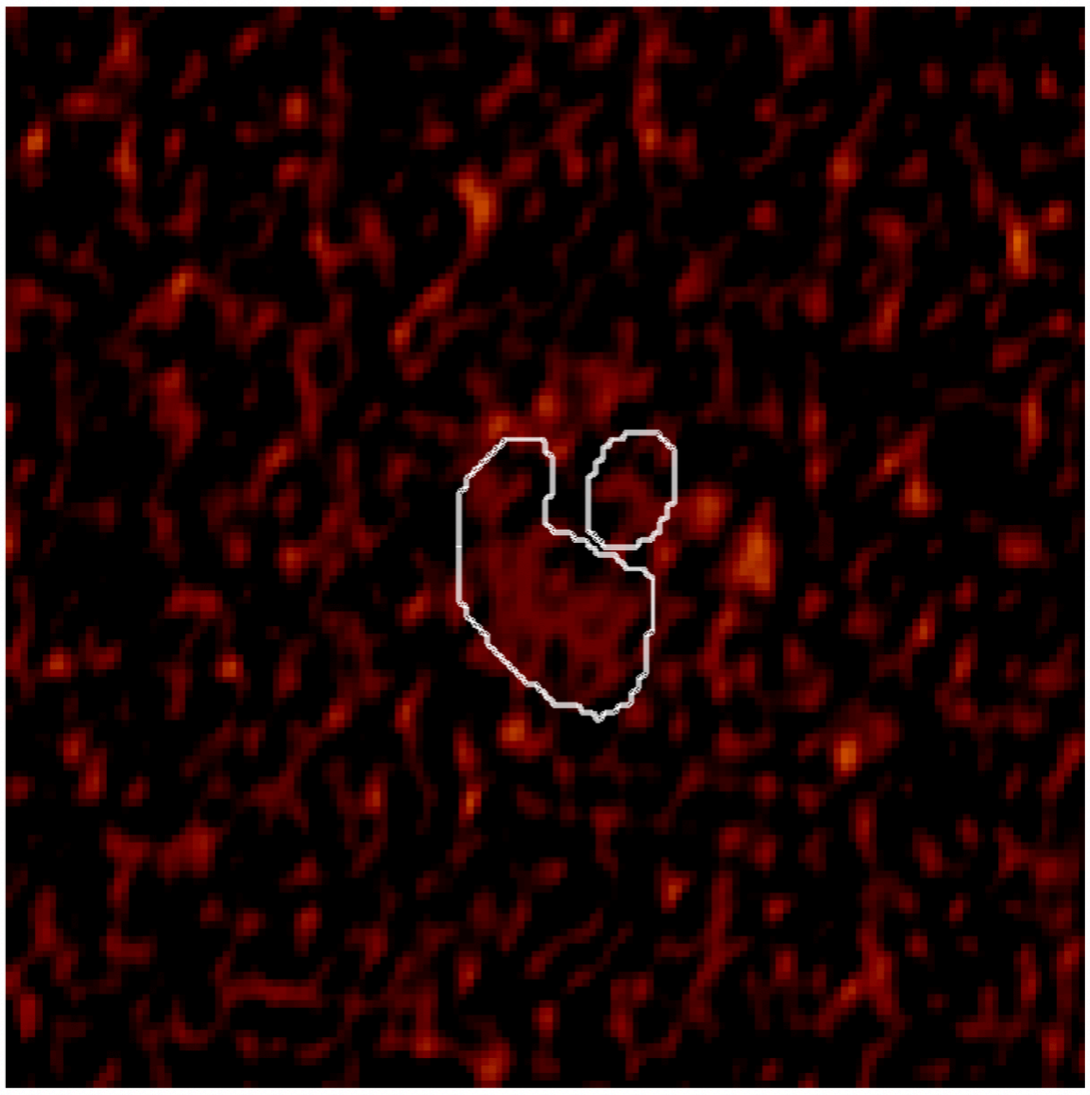


Automasking

Cleaned image (+ mask)



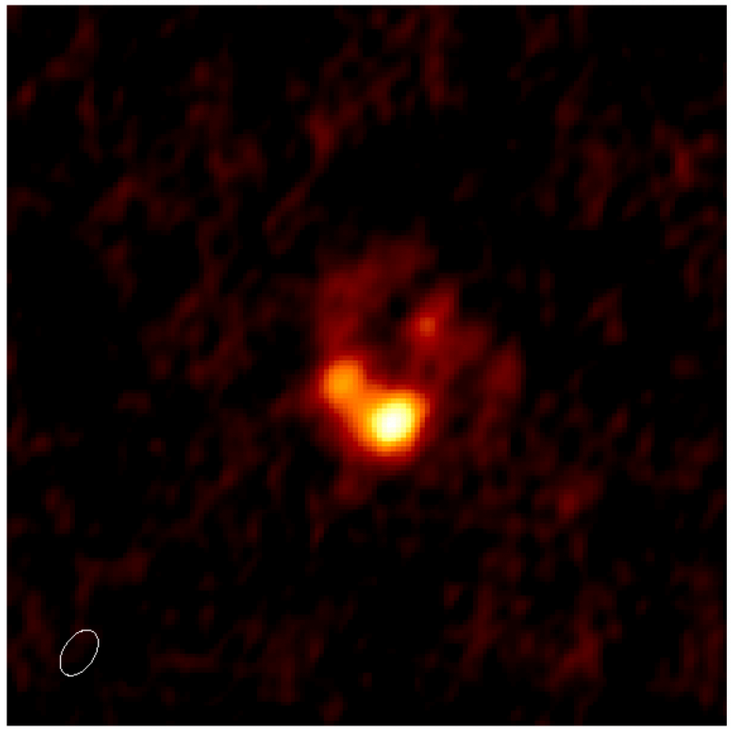
Residual (+mask)



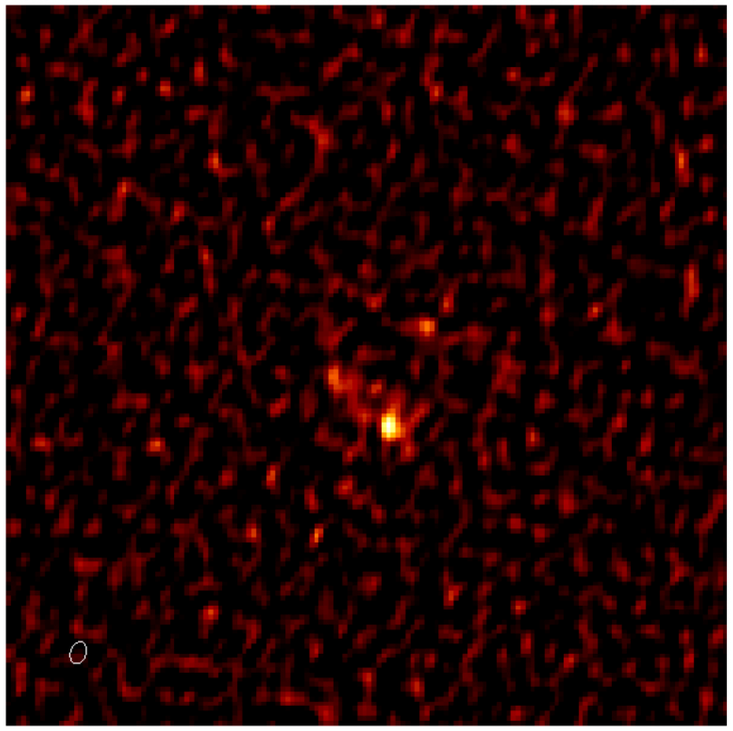
Weighting schemes

- Choice of weighting has a significant impact on the resultant image
- Trade-off between angular resolution and sensitivity
  - Need to decide what is most important for your science
  - Robust = 0.5 is often used — good compromise between the two

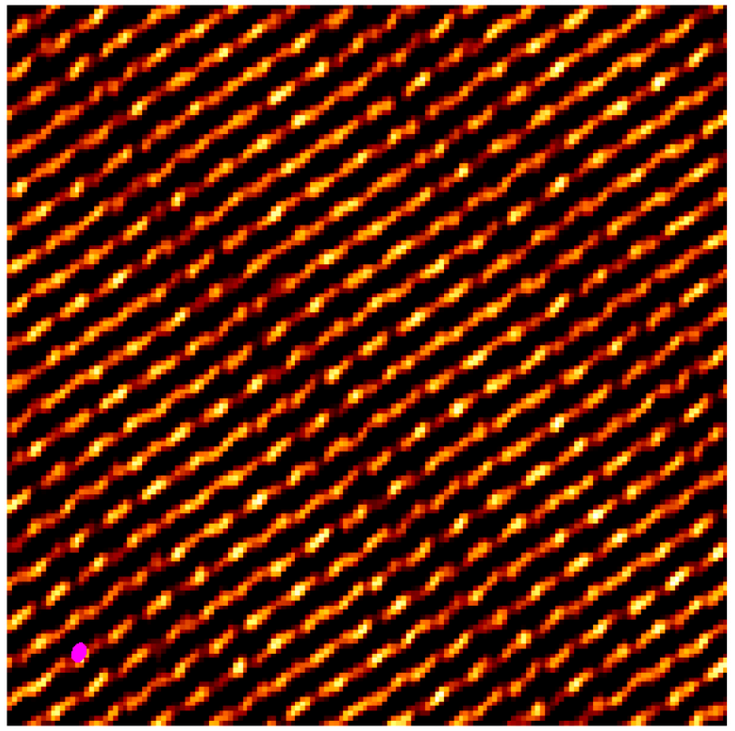
Natural (robust = 2)



Briggs (robust = 0)



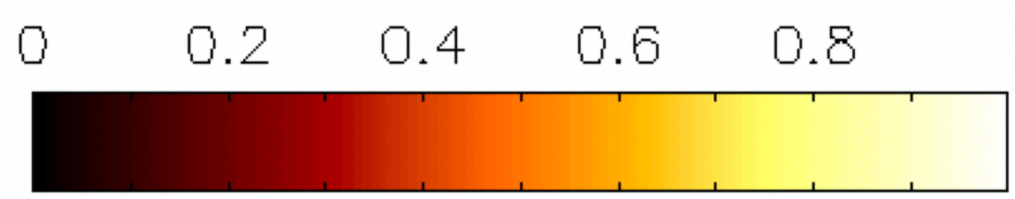
Uniform (robust = -2)



Primary beam correction

- Antenna response is not uniform across field of view — inherently noisier at the edges

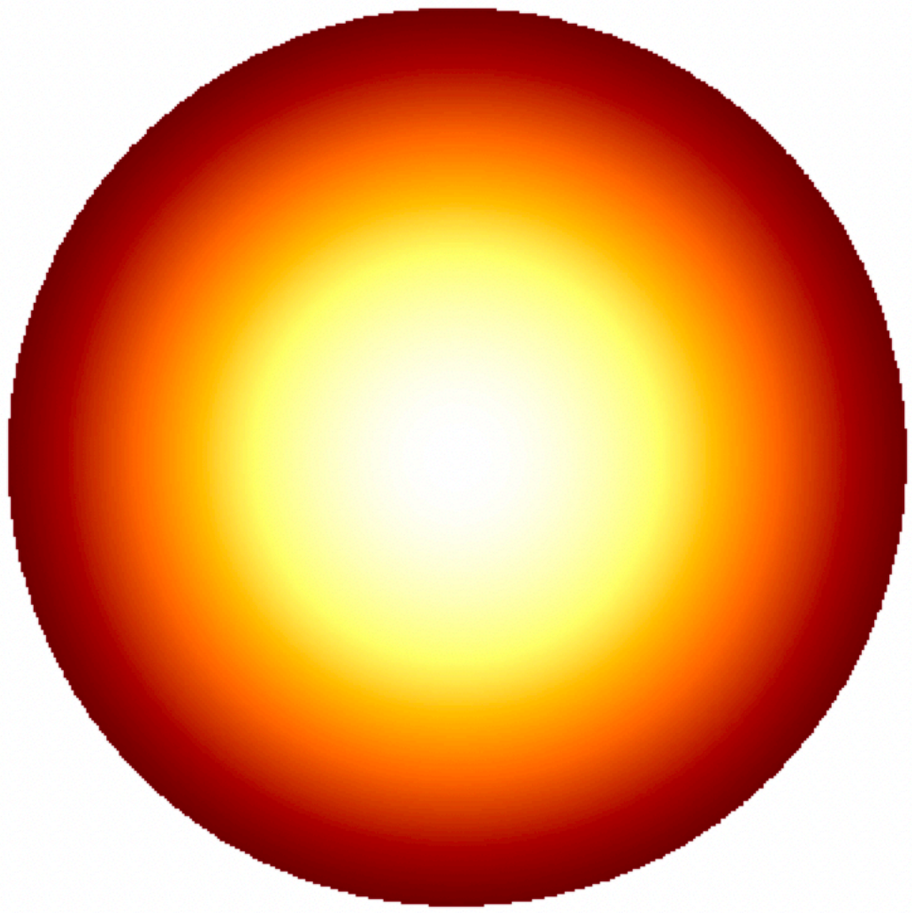
- This is not accounted for during imaging, and so we need to correct for it



- This can be done in two ways:

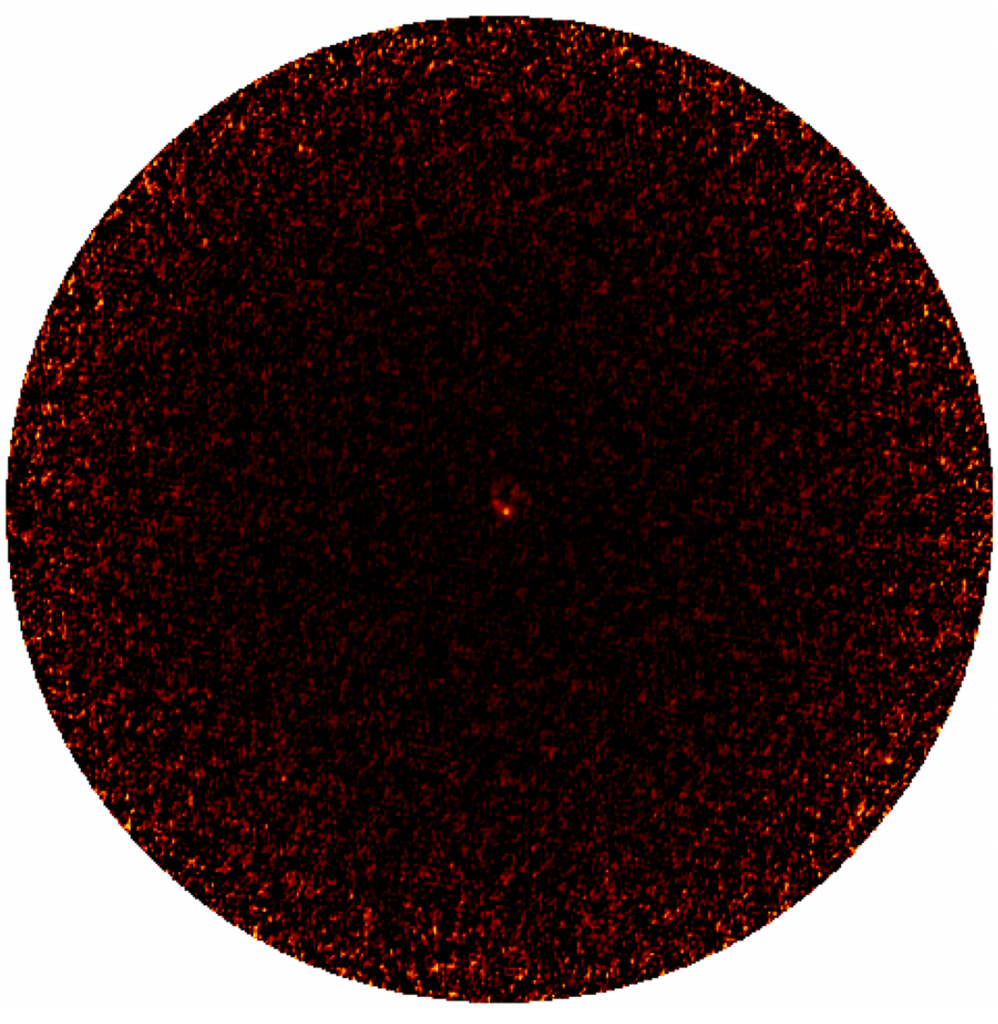
  - Setting `pbcor=True` during cleaning. This will output two images — with and without primary beam correction

  - Using the `impbcor` task post-cleaning

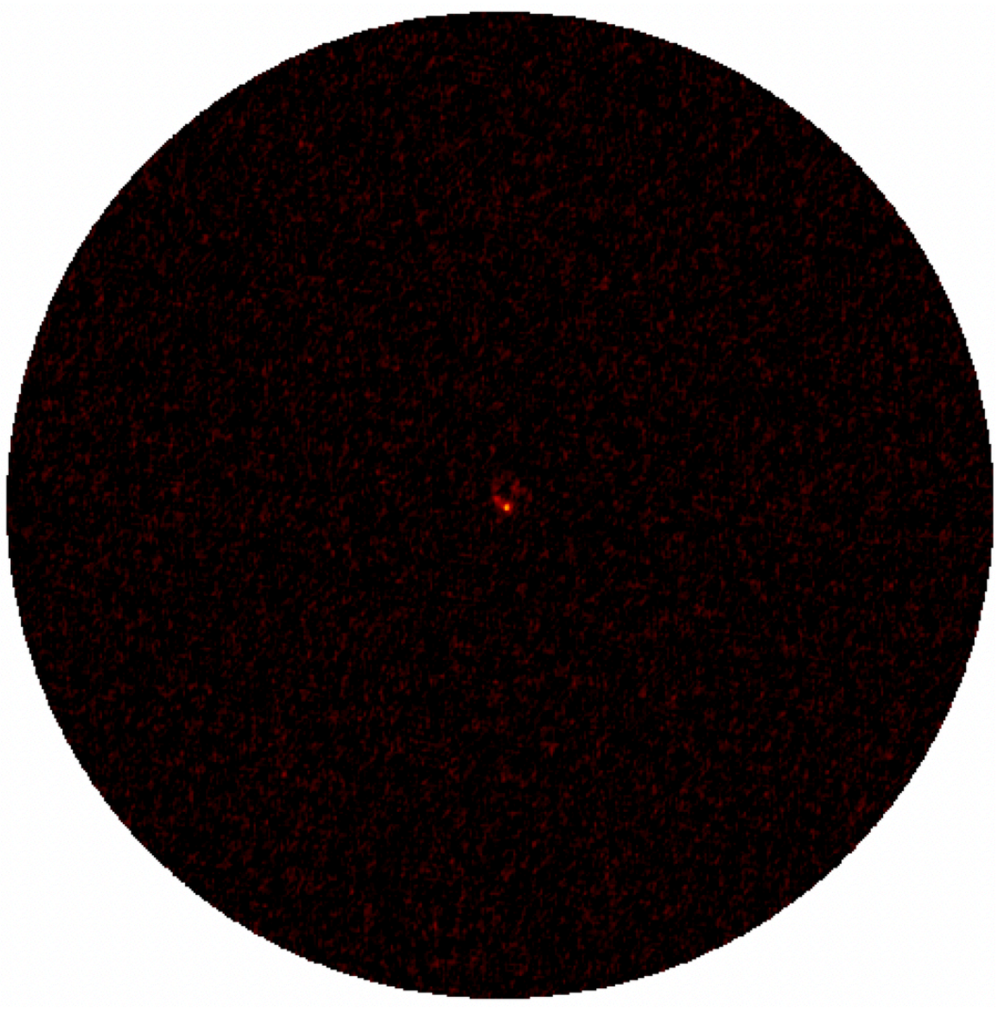


Primary beam correction

With PB-correction



Without PB-correction



Continuum imaging

- Let’s try full clean of the continuum using:
  - Cleaning threshold (based on RMS in dirty image)
  - Cropped image size (*optional*, purely for efficiency)
  - `pbcor = True` (to correct for primary beam response)
  - Masking
    - the script has auto-masking pre-filled, but I’d encourage you to delete these parameters, and set `interactive=True` to experiment with manual masking
  - A range of robust values
  - Plus any other parameters you want to play with!

Automasking parameters

`usemask = 'auto-multithresh'`

- `noisethreshold`: sets the noise threshold above which emission is masked during the initial round of mask creation

- `sidelobethreshold` = sets a threshold based on the sidelobe level, above which significant emission is masked during the initial round of mask creation

- `lownoisethreshold`: defines threshold into which the initial mask is expanded to include lower signal-to-noise regions

- `minbeamfrac`: minimum size a region must be to be retained in the mask (as a fraction of the beam size)

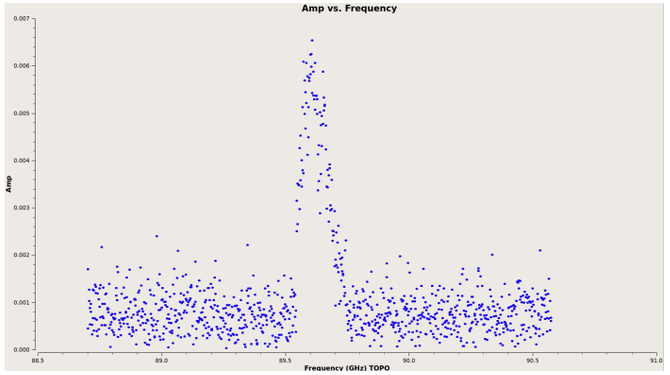
- `growiterations`: number of iterations per clean cycle for mask growth. A larger number will allow the mask to grow to capture fainter, more extended emission (if present), but may increase the processing time significantly

Note that either `noisethreshold` or `sidelobethreshold` is used depending on which threshold is higher.

Spectral line imaging

- The process of cleaning line data is broadly similar to cleaning the continuum, but there are some extra steps and parameters:
    - We now have a third dimension (frequency/velocity) made up of channels, the spacing of which is related to the spectral resolution of the data
    - The emission changes from channel-to-channel, which makes cleaning and masking more complex
    - More data → longer processing time
    - The dust continuum level must be subtracted to ensure that we are imaging only the line emission

Splitting out the CO data



- For this particular dataset there are four SPWs, only one of which contains spectral line emission (recall this plot from yesterday)
- This is CO (3-2) emission, which is contained in SPW 25
- To make the data more easily manageable, we can split out this SPW:

```
split(  vis                 = visfile,
        outputvis           = visfile+'.split',
        field               = 'PJ113921.7',
        spw                 = '25',
        datacolumn          = 'corrected')
```

Note: splitting out SPWs re-indexes them. In the output file there will only be one SPW, indexed as ‘0’

Continuum subtraction

- Having split out the CO SPW, we need to subtract the continuum emission. This can be done after imaging, but it's generally recommended to do this beforehand.

- Use CASA task `uvcontsub` to subtract the continuum from the *uv* data

```
uvcontsub(  vis                 = visfile+'.split',

            field               = 'PJ113921.7',

            spw                 = '0',

            fitspw              = contchans_CO,

            fitorder            = 1,

            excludechans        = False,

            want_cont           = False)
```

CO cube imaging

- Now we can start imaging the CO emission! Let's start by making a dirty image (0 clean iterations), just like we did for the continuum

```
    tclean(  vis             = visfile+'.split.contsub',
             imagename       = 'PJ113921.7.CO.cube.dirty',
             spw             = '0',
             restfreq        = '345.79599GHz',
             specmode        = 'cube',
             imsize          = [1250, 1250],
             cell            = '0.09arcsec',
             deconvolver     = 'hogbom',
             niter           = 0,
             weighting       = 'briggsbwtaper',
             robust          = 0.5,
             gridder         = 'standard',
             interactive     = False)
```

CO cube imaging

- As for the continuum, we see that the emission is concentrated to the inner portion of the field, so we can crop the image when cleaning
- We also see that the line emission only appears in a small portion of the channels, so we can restrict the imaging to just these channels
- Once again, we want to increase the number of clean iterations, add a cleaning threshold, and add auto masking parameters …

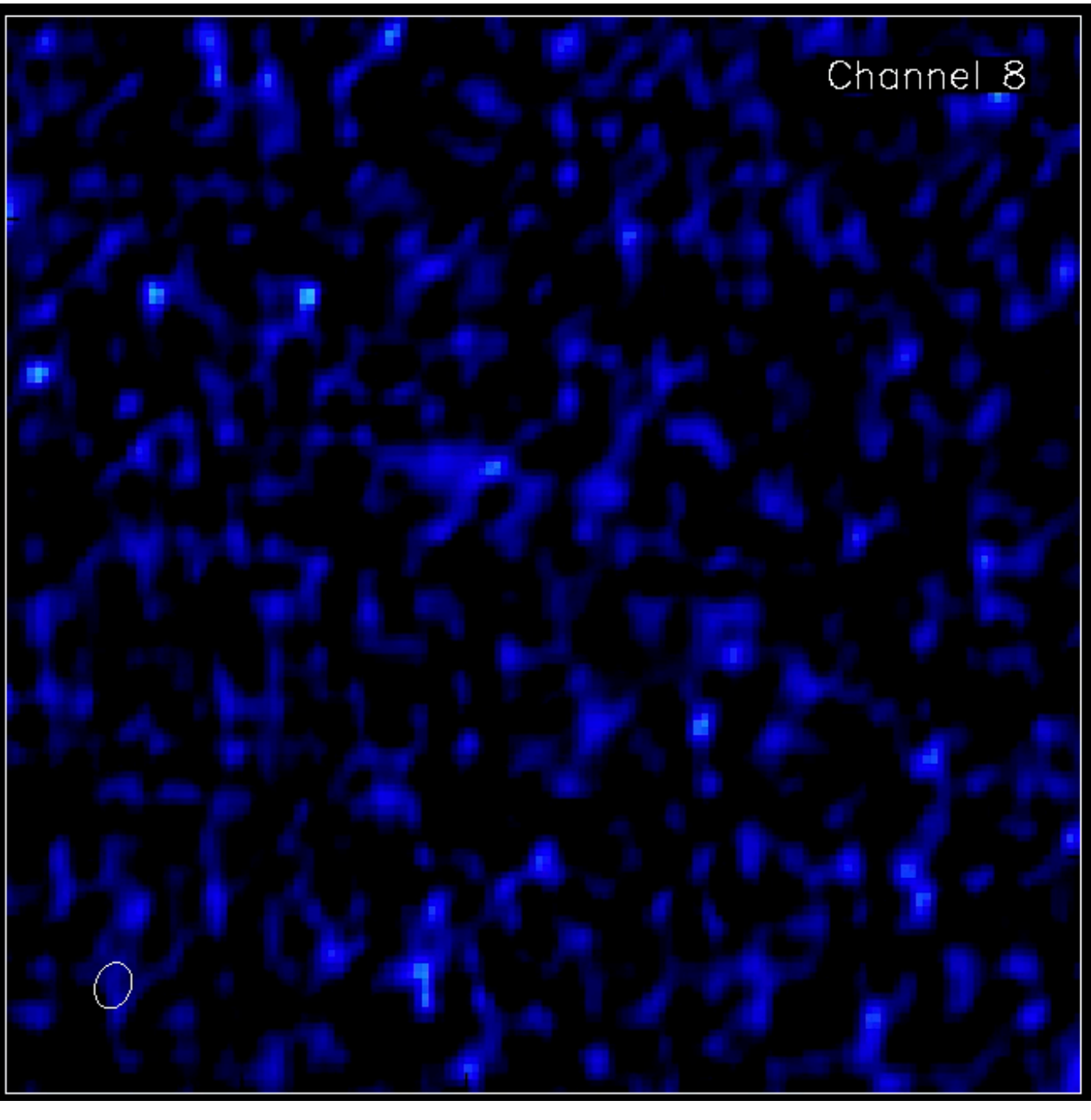
CO cube imaging

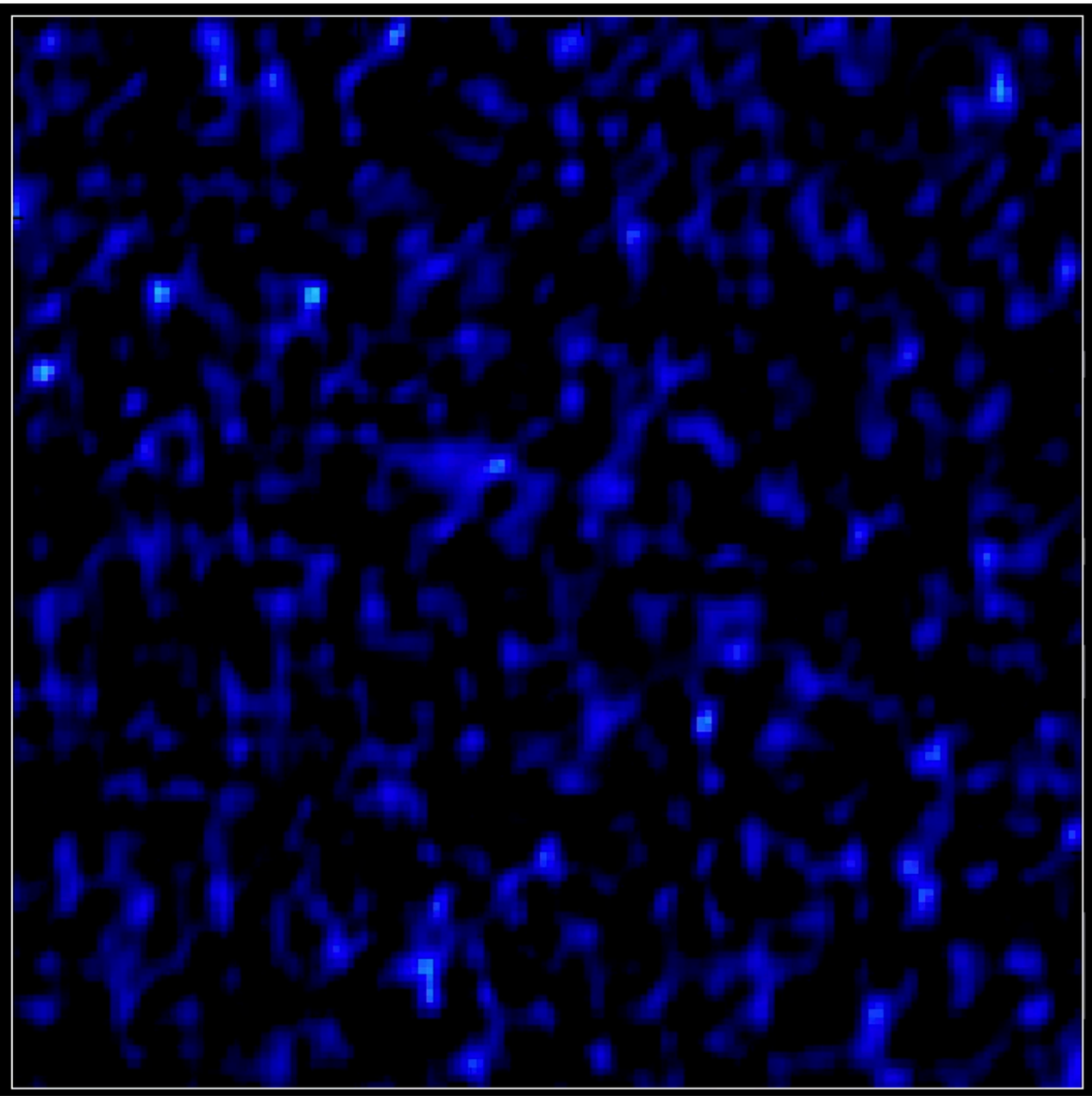
```
tclean(  vis                 = visfile+'.split.contsub',
         imagename           = 'PJ113921.7.CO.cube',
         phasecenter         = 'ICRS 11:39:21.728 20.24.51.838',
         spw                 = '0',
         restfreq            = '345.79599GHz',
         specmode            = 'cube',
         imsize              = [150, 150],
         cell                = '0.09arcsec',
         deconvolver         = 'hogbom',
         niter               = 1000000,
         weighting           = 'briggsbwtaper',
         robust              = 0.5,
         pbcor               = True,
         threshold           = '0.001Jy',
         nchan               = 100,
         start               = 200,
         width               = 1,
         interactive         = False,
         usemask             = 'auto-multithresh',
         sidelobethreshold   = 2.0,
         noisethreshold      = 4.0,
         lownoisethreshold   = 2.5)
```

CO cube imaging

Image

Residual

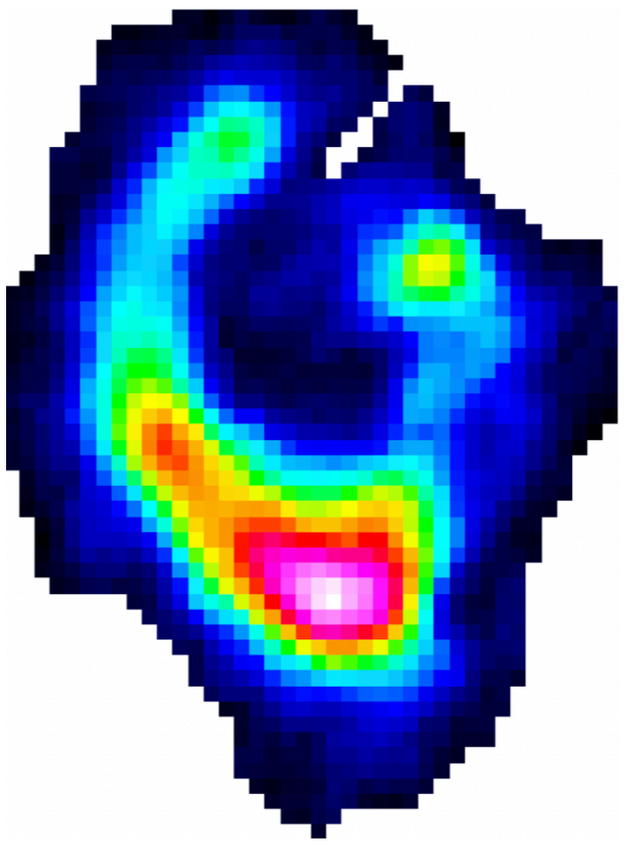




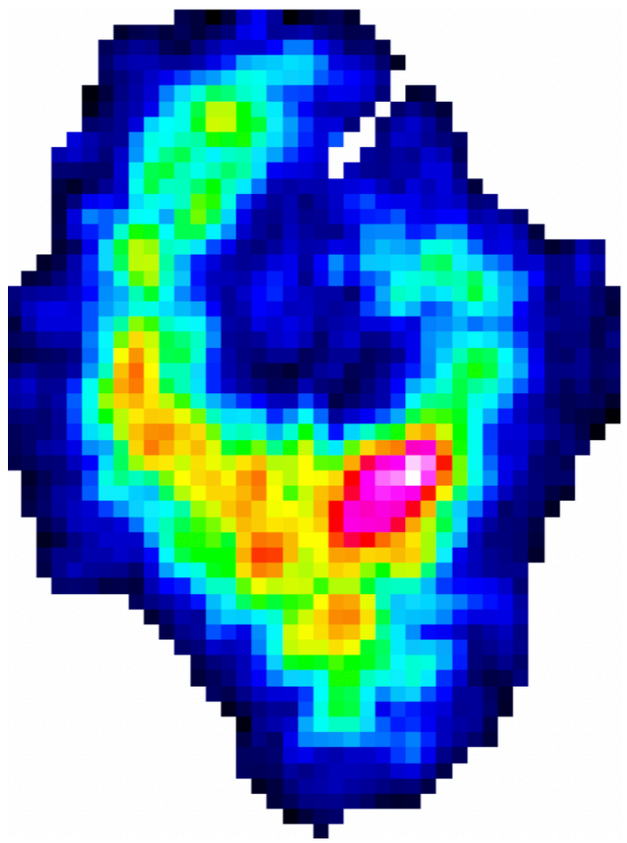
Moment maps

```
immoments(imagename     = 'PJ113921.7.CO.cube.image',
          moments       = [0, 1, 8],
          region        = 'moment_region.crtf',
          chans         = '10~55',
          includepix    = [0.0007, 100],
          outfile       = 'PJ113921.7.CO.cube.moment')
```

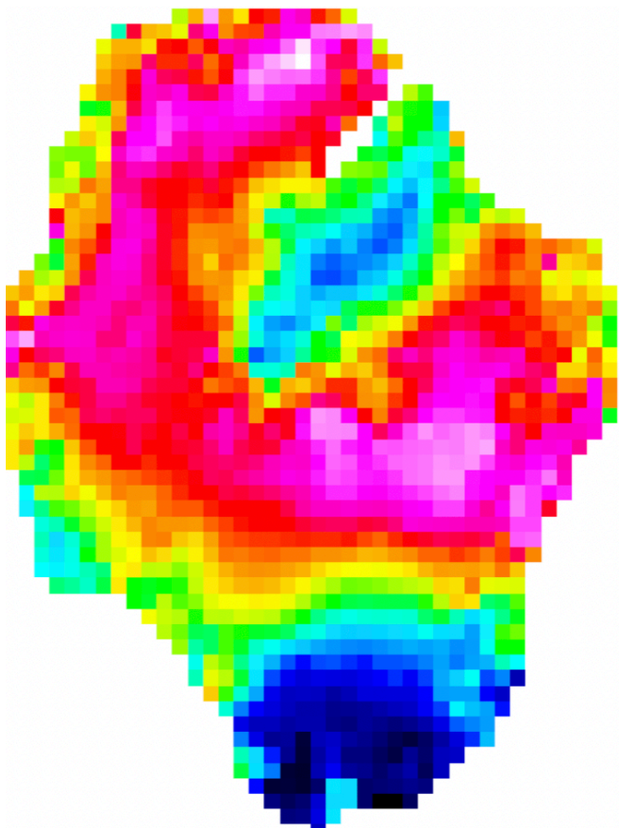
Integrated intensity (moment 0)



Maximum intensity (moment 8)



Velocity field (moment 1)



Alternative cube analysis tools

Spectral Cube (Python)

- Toolkit for reading, writing, manipulating, and analysing spectral cube data
- Create sub-cubes, moments, extract spectra etc.
- Designed to work with very large cubes that are too large to load into memory

Pyspeckit (Python)

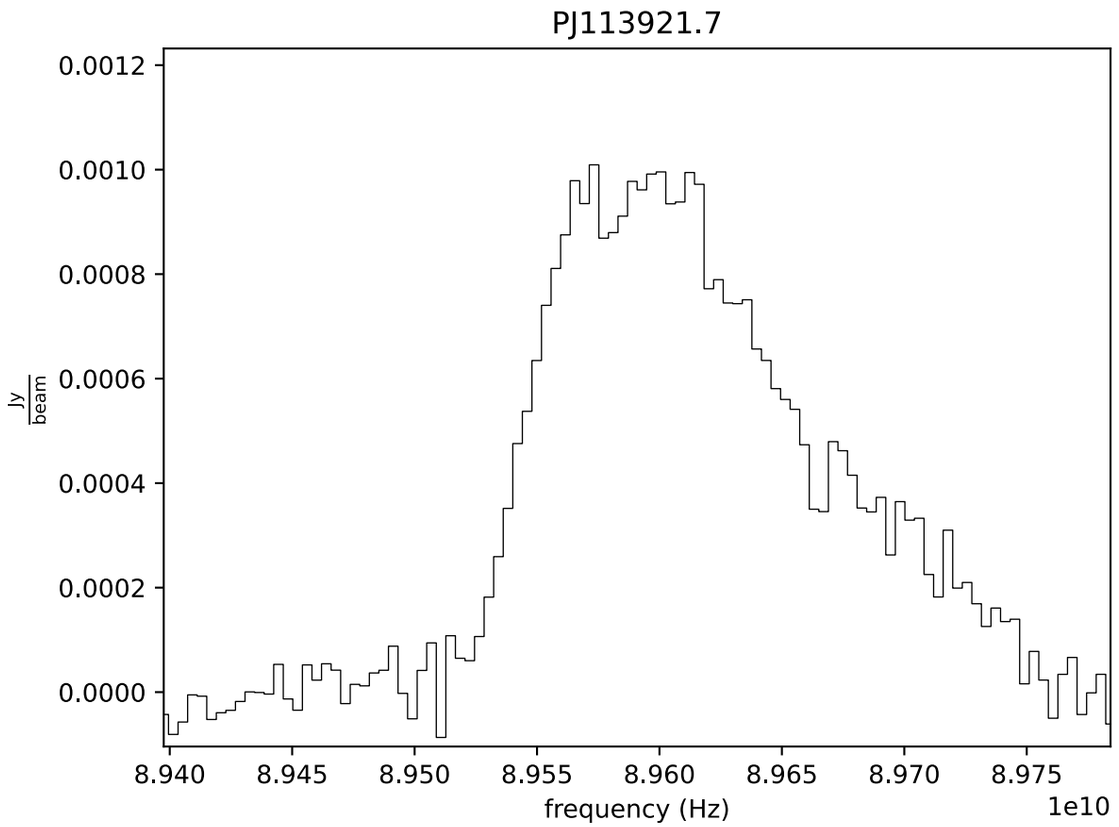
- Analysis toolkit for analysing spectra
- Plotting, line fitting, line modelling, and more

Alternative cube analysis tools

```
import os
import glob
import pyspeckit
import matplotlib.pyplot as plt
from spectral_cube import SpectralCube


filename = 'PJ113921.7.CO.cube.image.crop.fits'
cube     = SpectralCube.read(filename)
meanspec = cube.mean(axis=(1,2))
assert meanspec.size == cube.shape[0]
meanspec.write(filename.replace('.fits', '.meanspec.fits'))


sp = pyspeckit.Spectrum(filename.replace('.fits', '.meanspec.fits'))
sp.plotter(color = 'k', linestyle='-')
plt.tight_layout()
sp.plotter.savefig('mean_spec_crop.pdf')
plt.close()
```



Parallel processing (Linux only!)

You can run `tclean` in parallel across multiple cores in order to distribute the processing and speed things up:

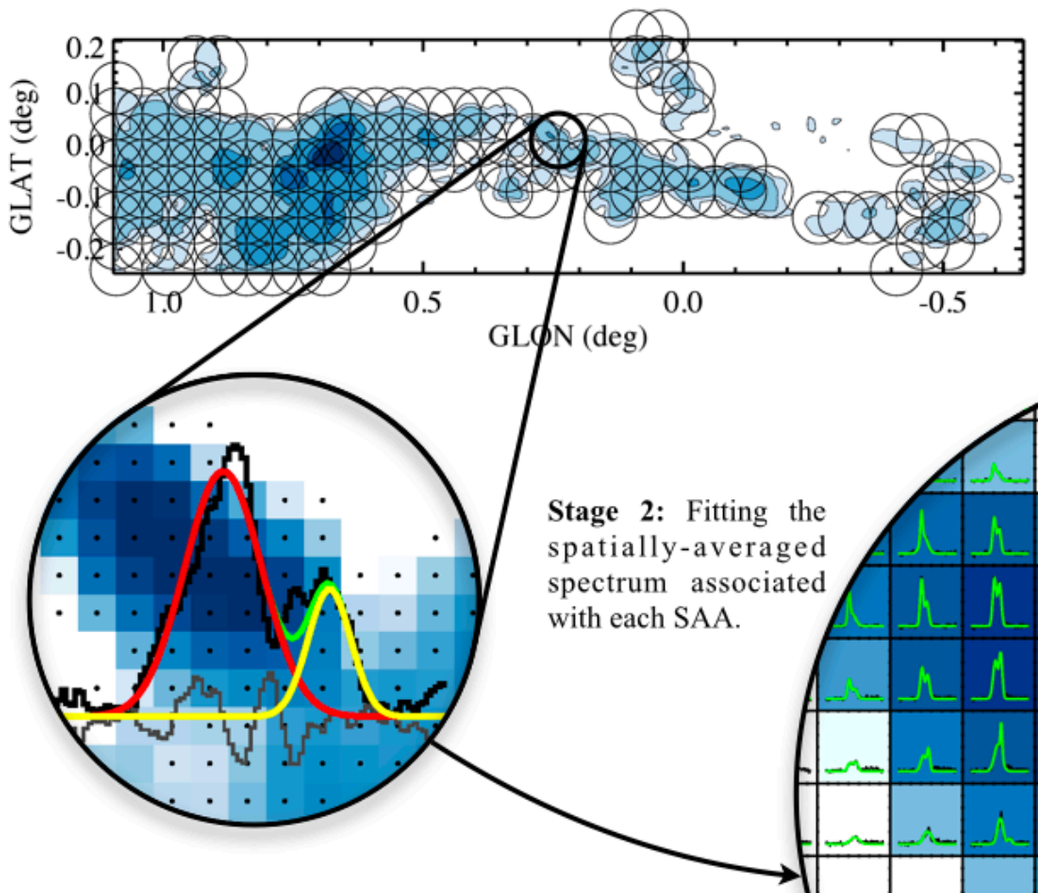
- In `tclean`, specify the parameter `parallel=True`
- Place your `tclean` command in a `.py` script
- Run your script via:

Number of cores

```
/path_to_casa/bin/mpicasa -n 8 /path_to_casa/bin/casa --
nologger -c ./imaging_script.py
```

[You can also place the above command into a .sh script and execute it in the background]

SCOUSE(py): Semi-automated multi-COmponent Universal Spectral-line fitting Engine



Stage 1: Identify the spatial area over which to implement SCOUSE - A grid of Spectral Averaging Areas (SAAs; black circles where 50% of the enclosed positions have a non-zero integrated intensity).

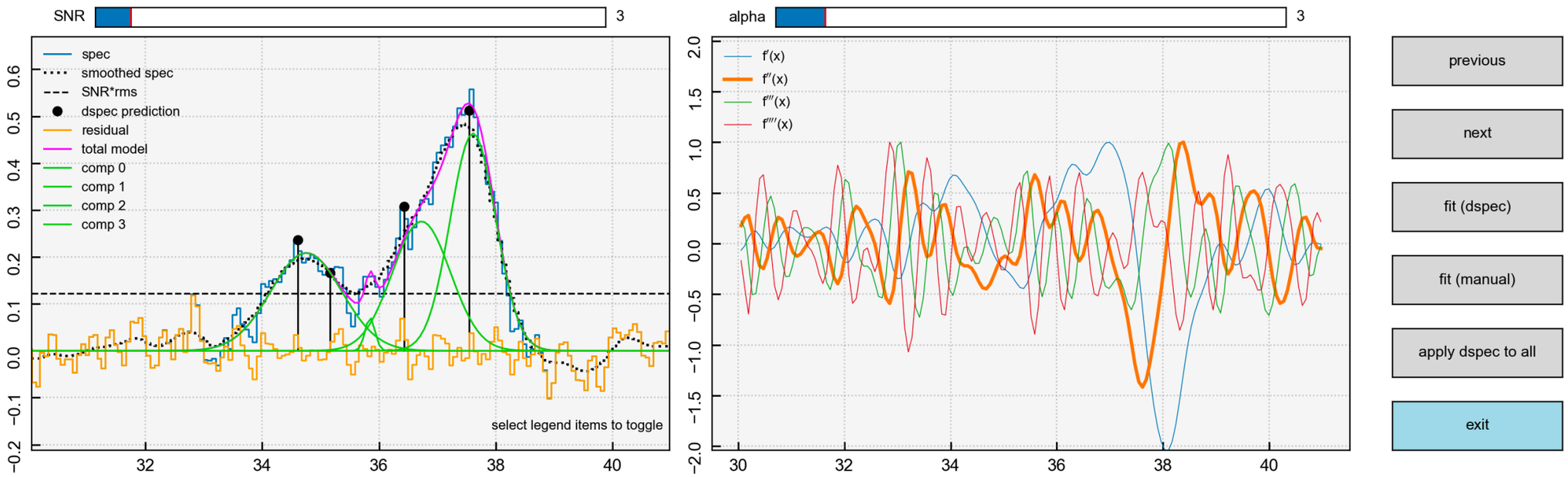
Stage 2: Fitting the spatially-averaged spectrum associated with each SAA.

Stages 3 & 4: Fitting the individual spectra using output parameters from stage 2 as free-parameter inputs, and selecting the "best-fits" to each spectrum.

Available for download: **https://scousepy.readthedocs.io/en/latest/**

For each averaged spectrum, scousepy provides an initial fit, which the user cycles through an interactive GUI to either accept or update the fit.

scousepy will then enter the fully automated stage, where it will take these averaged fits, and pass the parameters to fit the spectrum at every single pixel within each averaging area.



derivative spectroscopy information:
SNR: 3
alpha size: 3
pyspeckit fit information: **Fit has converged...**
number of components: 4
amplitude: 0.21 +/- 0.01; 0.07 +/- 0.04; 0.28 +/- 0.05; 0.46 +/- 0.12
shift: 34.75 +/- 0.05; 35.85 +/- 0.04; 36.73 +/- 0.24; 37.61 +/- 0.1
width: 0.61 +/- 0.06; 0.08 +/- 0.05; 0.5 +/- 0.18; 0.39 +/- 0.04
goodness of fit information:
chisq: 118.95
red chisq: 0.82
AIC: -1022.81

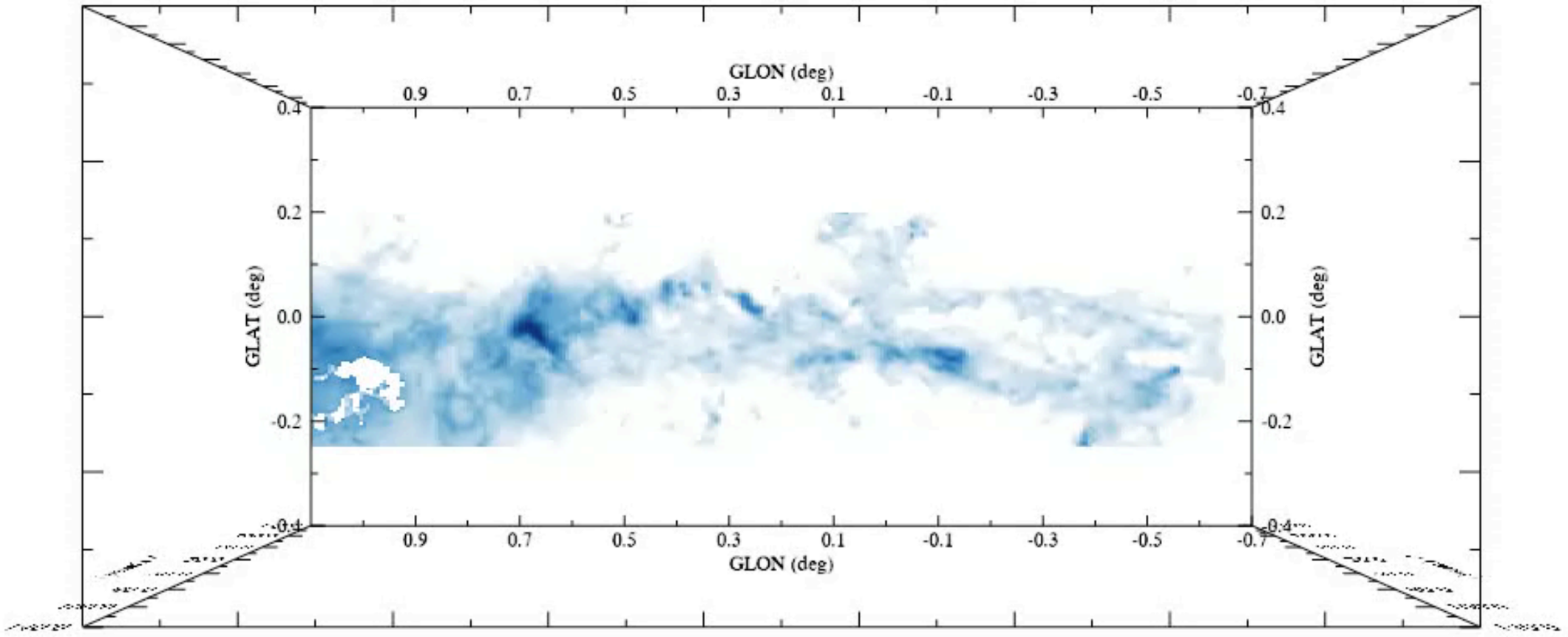
<< < 1 of 24 > >>

Once the automated fitting has completed, it will output an ascii file containing many measured parameters per pixel, e.g.:

- Number of components at that pixel location
- x
- y
- Amplitude
- Centroid velocity
- Velocity dispersion
- RMS

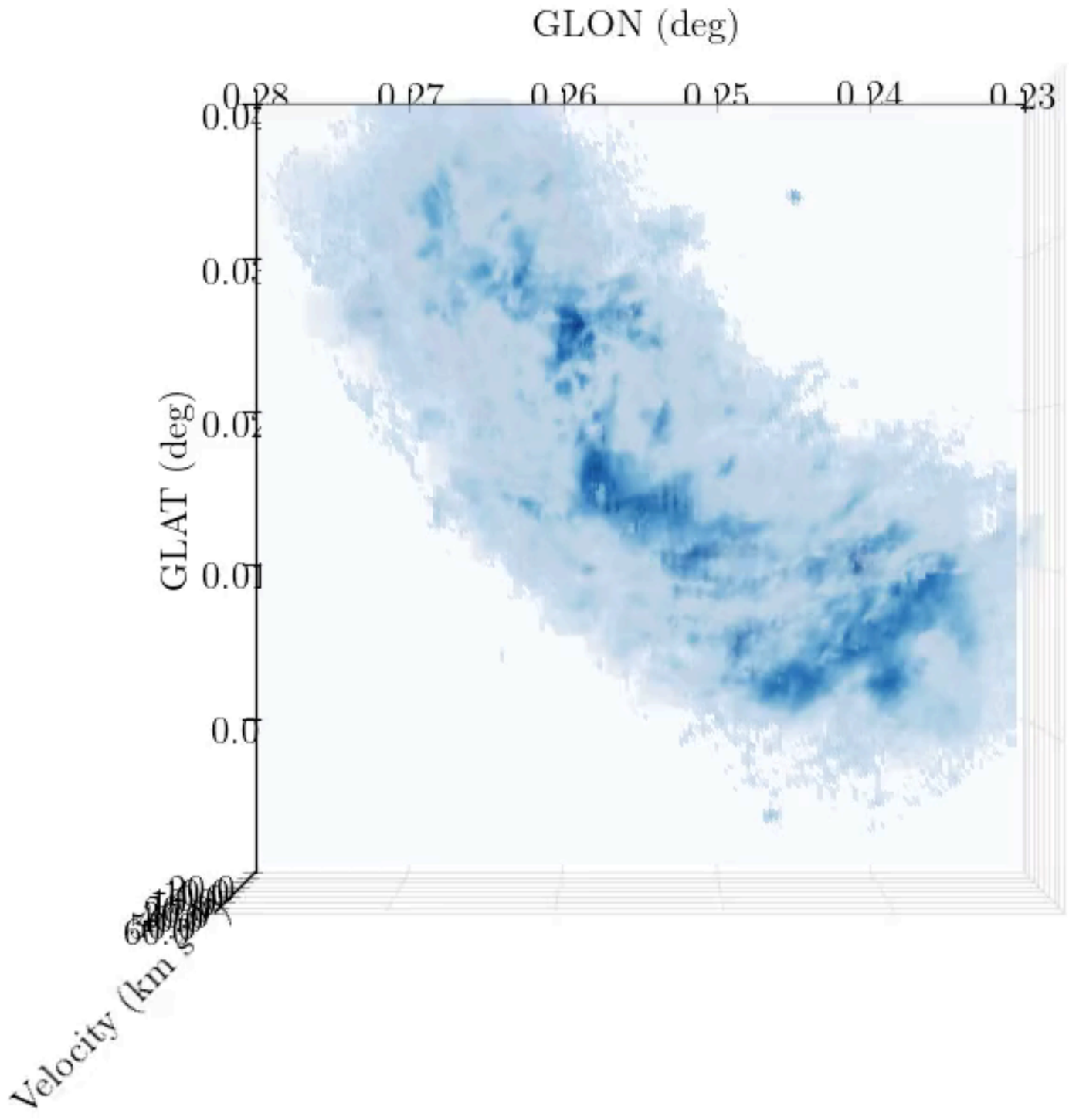
You can plot these data in various ways to make some cool plots …

SCOUSEpy …spectral line fitting of the inner ~250 pc of the Milky Way **Henshaw et al. 2016a**



“Large-scale” perspective; “Small-scale” detail

G0.253+0.016 (‘the Brick’, GC cloud)

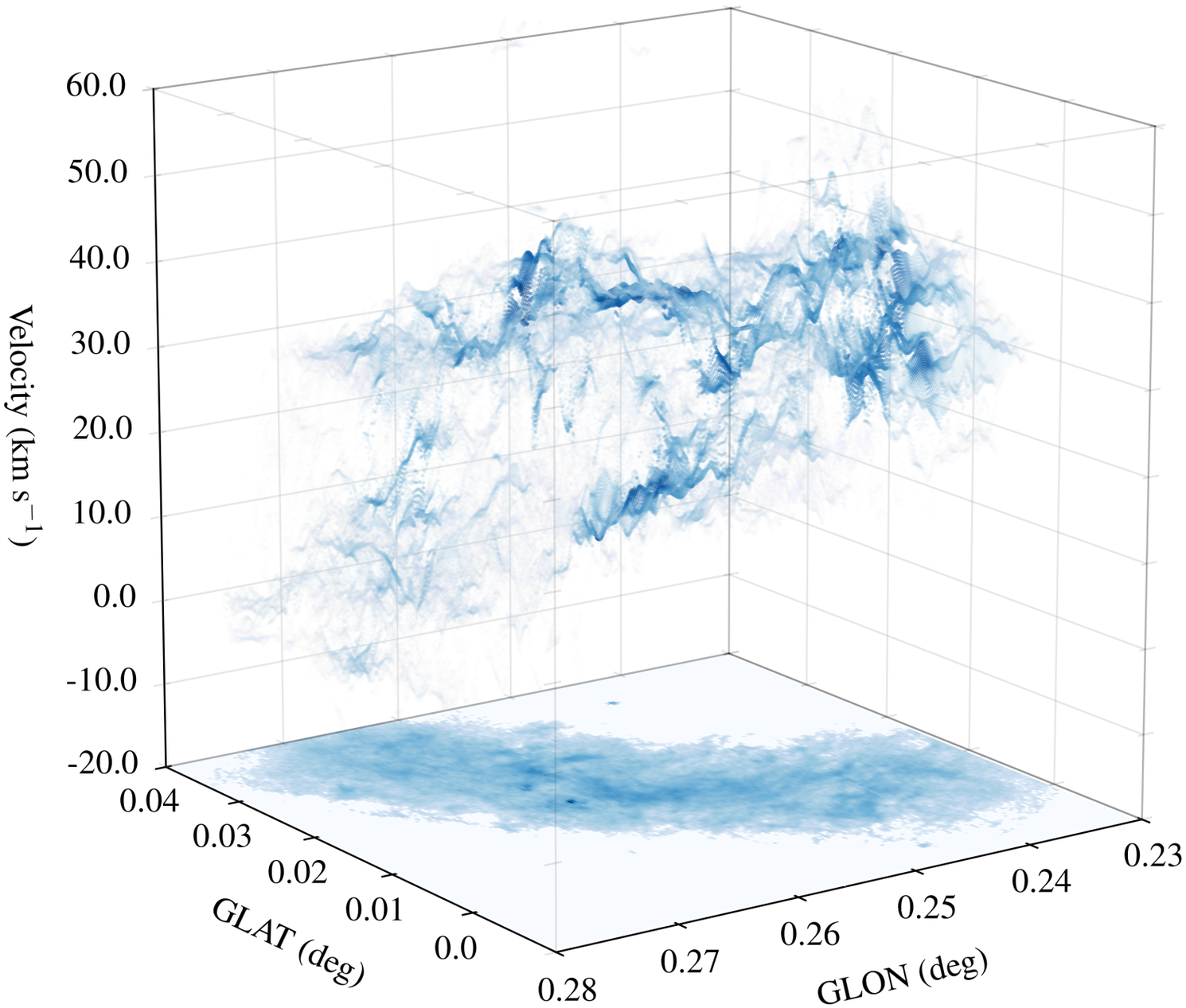


- HNCO @ 3mm
- 1.7” res ~ 0.07pc
- 3.4 km/s spectral resolution; 1.7km/s channels

Data courtesy of Rathborne et al. 2014b, 2015:
ALMA cycle 0: ALMA#2011.0.00217.S

SCOUSE's view of the Brick

Henshaw et al. 2019



The Brick is a **highly complex** and **sub-structured** molecular cloud, and velocity oscillations are ubiquitous

Henshaw et al. 2019

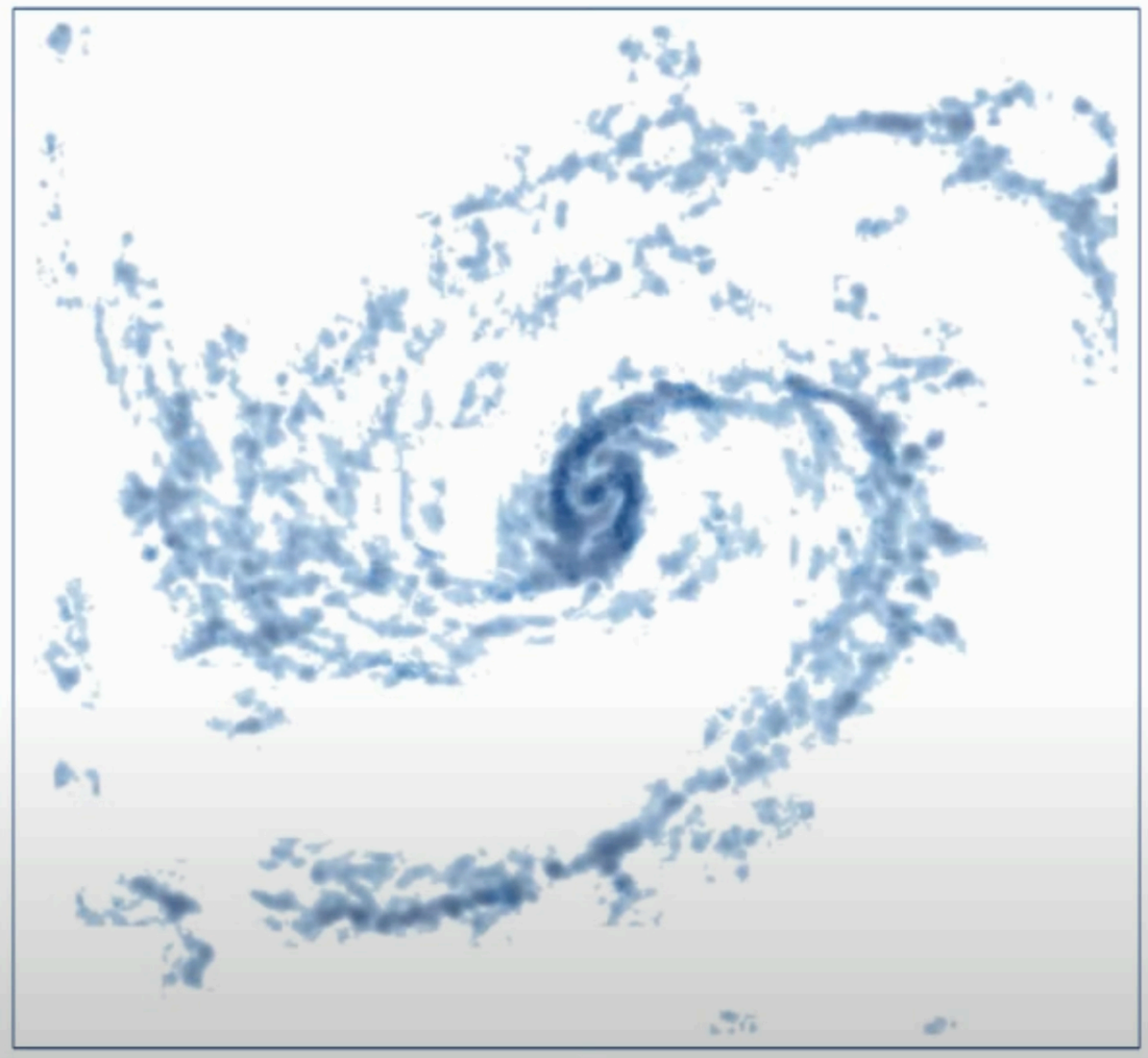
Ubiquitous Velocity Fluctuations throughout the Molecular Interstellar Medium

G0.253+0.016 a.k.a. "The Brick"

Jonathan D. Henshaw et al.

Henshaw et al. 2020

NGC 4321



as viewed on the plane of the sky (position-position view)

You can find a tutorial here if you fancy trying it yourself:

https://scousepy.readthedocs.io/en/latest/tutorial_v2.0.0.html