pymakeplots/CubeLineMoment Moment Calculation Comparison

Jeff Mangum (NRAO)

March 26, 2025

Abstract

This document describes a series of comparisons of the moment calculation results (mainly moment0) derived from using the CubeLineMoment[1] and the pymakeplots[2] packages.

1 Philosophy and Procedures

CubeLineMoment and pymakeplots follow two slightly different philosophies for generating moment images:

- CubeLineMoment uses a very simple approach to moment calculation. Provide CubeLineMoment with a spatial and spectral range over which to integrate, a signal level below which pixels are masked, and (optionally) a "tracer" spectral line that is used to guide the moment calculation toward a velocity range which the user believes should contain emission from the spectral line of interest.
- pymakeplots does more than just calculate moments. As Tim Davis states in his pymakeplots guide: "pymakeplots is a python module for making publication quality plots and output datafiles (including moment zero, one, two, PVD's and spectra) from interferometric datacubes." For its moment calculations, pymakeplots uses a "masked moment" approach (Dame, 2011). As described in Dame (2011), the masked moment technique suppresses noise in spectral line moment calculations. Apparently the moment masking technique was developed by Arnold Rots in the AIPS task MOMNT (Rots et al., 1990). The moment calculation procedure, as described by Dame (2011), starts with a spectral line data cube defined as $T(v, x, y)$ for which we want to produce a moment-masked version $TM(v, x, y)$ in which most of the emission-free pixels are blanked. Here v is velocity and (x, y) are the sky coordinates.
    1. Determine the rms noise in $T(v, x, y)$.

[1] https://github.com/keflavich/cube-line-extractor
[2] https://github.com/TimothyADavis/pymakeplots

2. Generate a smoothed version of the data cube $TS(v, x, y)$ by degrading the resolution spatially by a factor of 2 and in velocity to the width of the narrowest spectral lines generally observed.
3. Determine the rms noise in $TS(v, x, y)$. It is essential that the rms be measured accurately, either by using regions expected to be emission-free or by iteratively excluding regions of significant emission. Take care that your smoothing algorithm does not zero (rather than blank) edge pixels since this would artificially lower the rms. Likewise be aware of under-sampled regions that were filled by linear interpolation, since these will have higher rms in the smoothed cube.
4. Let the clipping level T_c equal N times the rms noise in the smoothed cube. A typical value for N is 5 (Dame (2011) provide a justification for using $T_c = 5$ in their use case). This step in the process is discussed further in Section 2.1.
5. Generate a masking cube $M(v, x, y)$ initially filled with zeros with the same dimensions as T and TS. The moment masked cube $TM(v, x, y)$ will be calculated as $M * T$.
6. For each pixel $TS(v_i, x_j, y_k) > T_c$, unmask (set to 1) the corresponding pixel in M.
7. Also unmask all pixels in M within the smoothing kernel of $TS(v_i, x_j, y_k)$, since all of these pixels weigh into the value of $TS(v_i, x_j, y_k)$. That is, unmask within nv pixels in velocity and within ns pixels spatially, where"

$$nv = \frac{0.5 * fwhm_v}{dv} \tag{1}$$

$$ns = \frac{0.5 * fwhm_s}{ds} \tag{2}$$

   Dame (2011) notes that this is a refinement of conventional moment masking techniques.
8. Calculate the moment-masked cube $TM(v, x, y) = M * T$.
9. Zeroth moment maps can be obtained by summing over any dimension of $TM(v, x, y)$.

It should also be noted that Adam Ginsburg has informed me that the ACES survey uses a masked-moment approach[3] for calculating moments.

2 Some Details

2.1 pymakeplots Clip Level T_c

To determine the clip level T_c to use in the pymakeplots calculation one needs to calculate moment 0 images using a range of multiples of the rms noise (σ_{TS}) in the smoothed cube $TS(v, x, y)$. Dame (2011) compared the resultant moment 0 images to a "noiseless" image that they created from their CO survey measurements. Rather than go to the effort of creating a noiseless image, though, it is usually good enough to simply examine the moment 0 images created with varying clipping

[3] https://github.com/ACES-CMZ/reduction_ACES/blob/e988851840d8b19786f514e015b72432c5871051/aces/analysis/giantcube_cuts.py

steps N. For the use case described in Dame (2011) a value of $N = 5$ was found to be optimal. For the tests described below, larger values of N were found to work well.

Adam Ginsburg notes that masked-moment approaches are definitely the way to go for single spectral line analyses. They might also be the right approach for establishing the "reference" map in CubeLineMoment. One advantage of using the reference transition, rather than doing an expanded mask approach (as masked-moment calculations do) for each transition, is that you are always comparing the same velocities to each other for the target and reference transitions. In essence, there are two reasons to use CubeLineMoment:

- Contamination. If you use a pure masked-moment approach, but the negative velocity of one line is in the same channel as the positive velocity of another line somewhere else in the galaxy, you're screwed.
- Comparing apples to apples. Integrating over 'the complete spectral line' can mean that you're comparing different clouds entirely. CubeLineMoment at least ensures they are the same spatio-spectral pixels.

3 pymakeplots/CubeLineMoment Comparison Tests

The data for this experiment were derived from the "high-resolution ALCHEMI" project where imaging at $\sim 0.4''$ within ALMA Bands 6 and 7 were collected toward the NGC 253 CMZ. The moment 0 image calculations for CubeLineMoment used the following YAML file:

```
cube: /Users/jmangum/Science/ALCHEMI/ALCHEMIHighResBand67/Imaging/ngc253_h_06_TM1
      /spw29/uid___A001_X133d_X1579.ngc253_sci.spw29.cube.selfcal.I.fits
cuberegion: CubeLineMoment.reg
cutoutcube: /Users/jmangum/Science/ALCHEMI/ALCHEMIHighResBand67
           /Imaging/ngc253_h_06_TM1/spw29
           /uid___A001_X133d_X1579.ngc253_sci.spw29.cube.selfcal.I.fits
cutoutcuberegion: CubeLineMoment.reg
vz: 236
target: NGC253
brightest_line_name: HCN_32
brightest_line_frequency: 265.88618
width_line_frequency: 265.88618
velocity_half_range: 250
noisemapbright_baseline: [[55,60],[150,165]]
noisemap_baseline: [[55,60],[150,165]]
my_line_list: 265.88618
my_line_widths: 120
my_line_names: HCN_32
signal_mask_limit: None
spatial_mask_limit: None
```

```
mask_negatives: False
sample_pixel: Region6.reg
use_default_width: True
min_width: 10
min_gauss_threshold: 0.10
max_gauss_threshold: 0.90
```

...while for HCN $4-3$ the YAML file was a follows:

```
cube: /Users/jmangum/Science/ALCHEMI/ALCHEMIHighResBand67/Imaging/ngc253_m_07_TM1/spw25/uid
cuberegion: CubeLineMoment.reg
cutoutcube: /Users/jmangum/Science/ALCHEMI/ALCHEMIHighResBand67/Imaging/ngc253_m_07_TM1/spw
cutoutcuberegion: CubeLineMoment.reg
vz: 236
target: NGC253
brightest_line_name: HCN_43
brightest_line_frequency: 354.5054773
width_line_frequency: 354.5054773
velocity_half_range: 250
noisemapbright_baseline: [[15,20],[70,76]]
noisemap_baseline: [[15,20],[70,76]]
my_line_list: 354.5054773
my_line_widths: 130
my_line_names: HCN_43
signal_mask_limit: None
spatial_mask_limit: None
mask_negatives: False
sample_pixel: Region6.reg
use_default_width: True
min_width: 10
min_gauss_threshold: 0.10
max_gauss_threshold: 0.90
```

For pymakeplots the settings can be found in Tim Davis' pymakeplots tutorial at https://github.com/TimothyADavis/pymakeplots/blob/master/pymakeplots/quickstart/pymakeplots_tutorial.ipynb for instructions. Tim used `plotter.rmsfac=12` to generate the HCN $3-2$ and $4-3$ moment 0 images. For the HCN $4-3$ moment0 image Tim's pymakeplots script was as follows:

```
from pymakeplots import pymakeplots

path_to_flat_cube="uid___A001_X133d_X15b5.ngc253_sci.spw25.cube.selfcal.I.fits"

plotter=pymakeplots(cube=path_to_flat_cube)
plotter.obj_ra,plotter.obj_dec=11.888002,-25.288220
```

```
plotter.gal_distance=3.27
plotter.vsys=225
plotter.imagesize=10
plotter.chans2do=[26,70]

plotter.rmsfac=12
plotter.maxvdisp = 100.
plotter.posang=50.0
plotter.pvdthick=5

#plotter.make_moments(mom=[0,1,2],pdf=True,fits=True)
#plotter.make_pvd(pdf=False,fits=False)
#plotter.make_spec(pdf=True)
plotter.make_all(pdf=True,fits=True)
```

For the default test the comparisons between CubeLineMoment and pymakeplots were done using the following files:

- HCN $3-2$:
  - *ngc253_HCN32_mom0.fits*: pymakeplots moment0 image (from Tim's algorithm)
  - *NGC253_HCN_32_moment0_widthscale1.0_sncut999.0_widthcutscale1.0.fits*: CubeLineMoment image
  - *NGC253_HCN32_Moment0Ratio.fits*: Ratio of pymakeplots/CubeLineMoment
- HCN $4-3$
  - *ngc253_HCN43_mom0.fits*: pymakeplots moment0 image (from Tim's algorithm)
  - *NGC253_HCN_43_moment0_widthscale1.0_sncut999.0_widthcutscale1.0.fits*: CubeLineMoment image
  - *NGC253_HCN43_Moment0Ratio.fits*: Ratio of pymakeplots/CubeLineMoment

4 Results

4.1 Initial Tests

Figures 1, 2, 3, and 4 show the two moment0 images for each transition displayed using CARTA. It is clear that for both the HCN $3-2$ and $4-3$ comparisons that the two moment 0 image calculations differ. It appears that in both cases the pymakeplots images include more low-level extended emission than that derived from the CubeLineMoment images. To quantify the overall differences in each comparison we have calculated the ratio pymakeplots/CubeLineMoment (bottom-left panel in each figure). We also show in both figures (bottom-left) the pixel ratio histogram of

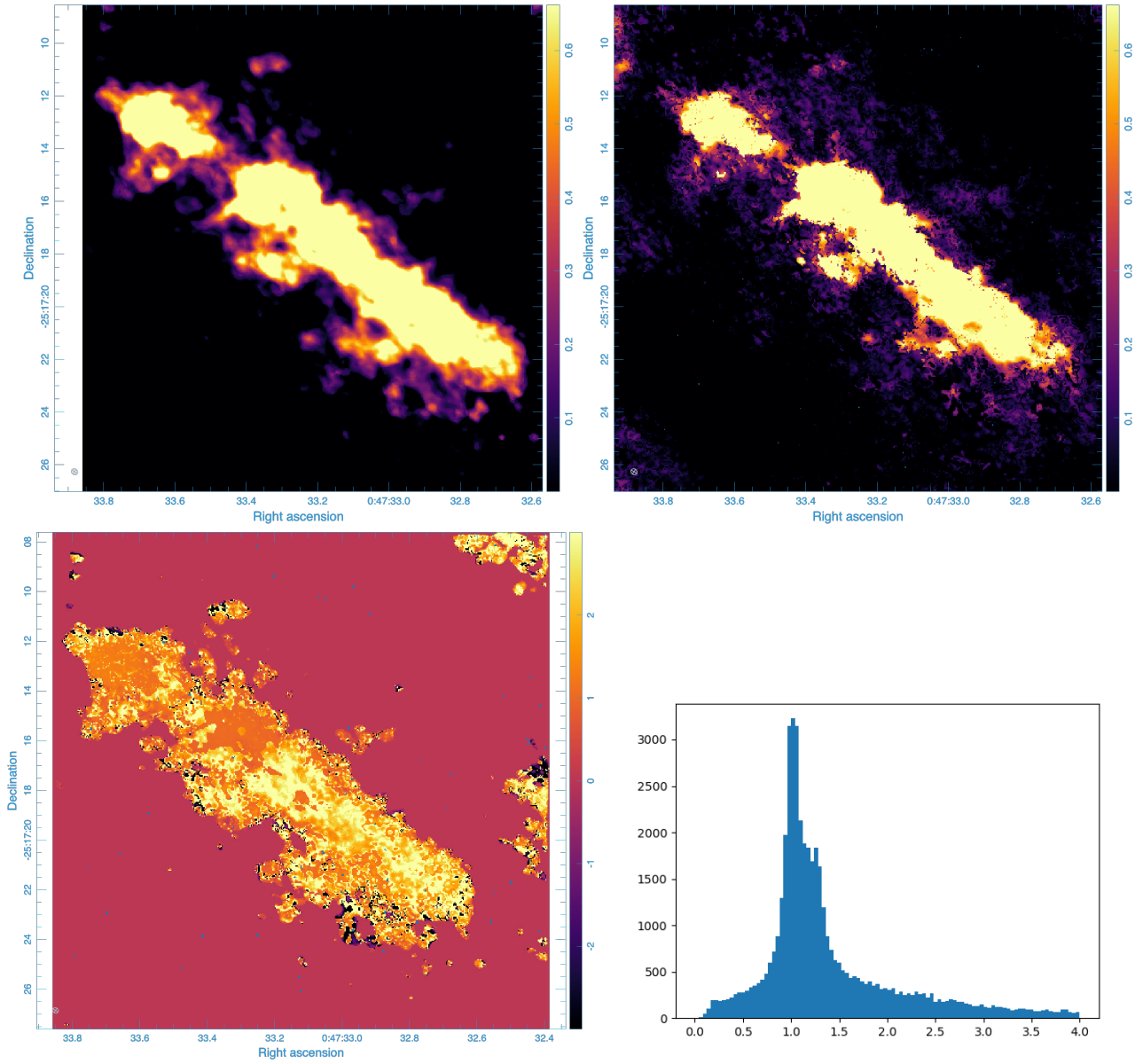

Figure 1: HCN $3-2$ moment 0 images derived from (top-left) pymakeplots and (top-right; min/max_gauss_threshold = 0.10/0.90) CubeLineMoment. Both moment 0 images are aligned both spatially and in intensity (Jy/beam*km/s). HCN $3-2$ moment 0 image ratio (bottom-left: pymakeplots/CubeLineMoment) and the associated ratio pixel histogram distribution (bottom-right, for ratios > 0.001). The intensity scale for the ratio image runs from -3 to 3 in the ratio.

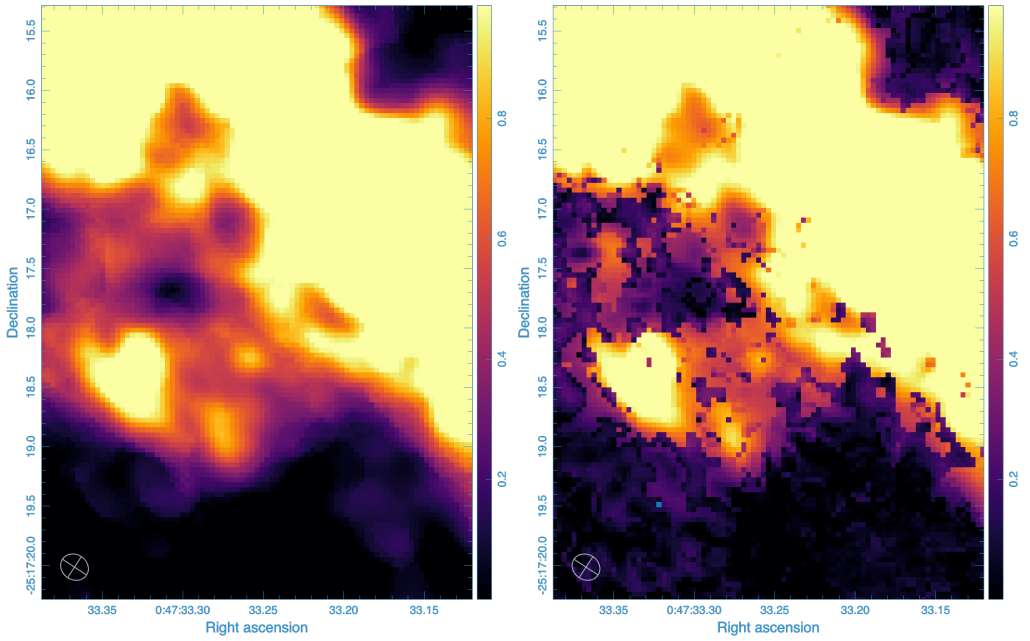

Figure 2: Zoomed version of the HCN $3-2$ moment 0 images derived from (left) pymakeplots and (right) CubeLineMoment (top row of Figure 1).

the ratio image. It seems pretty clear that indeed the pymakeplots moment 0 image includes more low-level (extended) emission.

It appeared that further investigation of the CubeLineMoment calculation parameters was in order. We did a series of tests where the CubeLineMoment input parameters min_gauss_threshold and max_gauss_thrshold were varied to see if there was any dependence of these CubeLineMoment parameters on the resultant pymakeplots/CubeLineMoment total flux ratio. Table 1 lists the results of these calculations and Figure 5 shows the respective pixel ratio distributions. Calculating the weighted sum of the ratio pixel distributions ($np.sum(ma)/len(ma)$, for ratios > 0.001) results in the values listed in the second column of Table 1.

4.2 After Error in Masking Fix

Adam discovered an error in the masking process on 2025-02-28 where a

```
threshold[threshold > min_gauss_threshold] = max_gauss_threshold
```

rather than

```
threshold[threshold > max_gauss_threshold] = max_gauss_threshold
```

was applied. This likely resulted in the odd results shown in Section 4.1. After Adam fixed this bug the HCN $3-2$ (and one of the HCN $4-3$) tests done in Section 4.1 were repeated, with the results

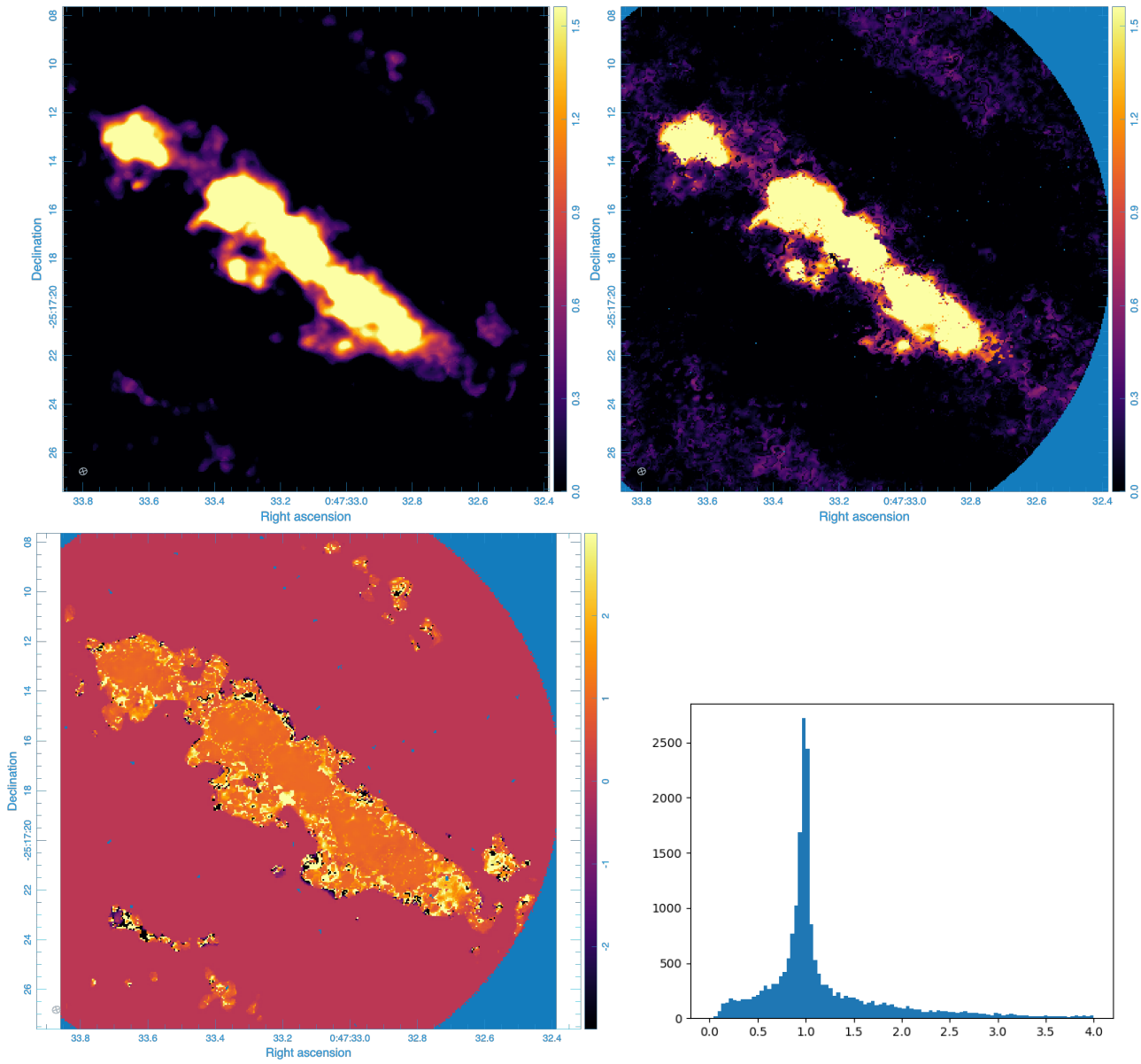

Figure 3: HCN $4-3$ moment 0 images derived from (top-left) pymakeplots and (top-right) CubeLineMoment (with min_gauss_threshold and max_gauss_threshold set to 0.10 and 0.90, respectively). Both moment 0 images are aligned both spatially and in intensity (Jy/beam*km/s). HCN $4-3$ moment 0 image ratio (bottom-left: pymakeplots/CubeLineMoment) and the associated ratio pixel histogram distribution (bottom-right, for ratios > 0.001). The intensity scale for the ratio image runs from -3 to 3 in the ratio.

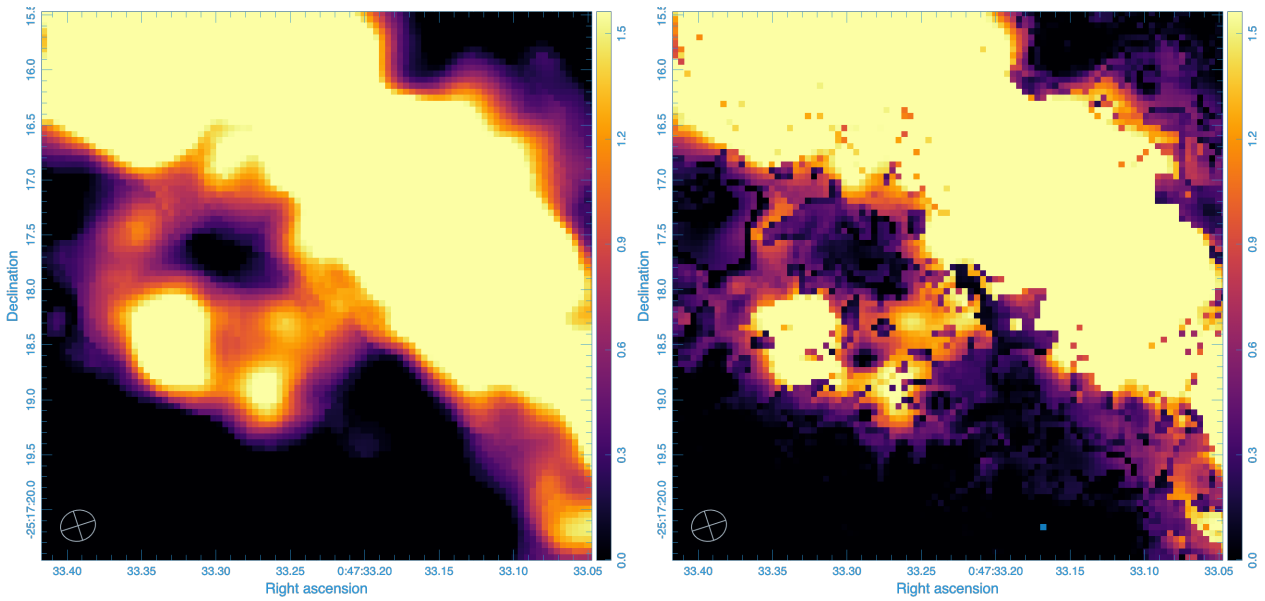

Figure 4: Zoomed version of HCN $4-3$ moment 0 images derived from (left) pymakeplots and (right) CubeLineMoment (top row of Figure 3).

shown in Table 2 and Figure 6. This masking bug was clearly producing the anomalous results listed in Section 4.1. A zoomed comparison of the HCN $3-2$ and $4-3$ moment0 calculations from pymakeplots and CubeLineMoment with the masking fix applied in shown in Figures 7 and 8, respectively.

4.3 Proper Comparison of pymakeplots and CubeLineMoment Moment0 Images

Note the proviso regarding the moment0 ratio calculations above: **All ratio calculations and histograms were calculated for ratio values greater than 0.001**[4]**.** Since CubeLineMoment produces a moment0 image with noise while pymakeplots does not, limiting their ratio to positive values greater than 0.001 seems to be a correct comparison.

4.4 Adding Dilation to Smooth-Out Pixel Gaps in CubeLineMoment Moment0 Images

Adam added **dilation** options to CubeLineMoment (YAML file entries):

- *dilation_iterations: 4*
- *erosion_iterations: 2*

[4]I also tried using a small value of 0.0001 and found that this made no difference to the weighted sum value.

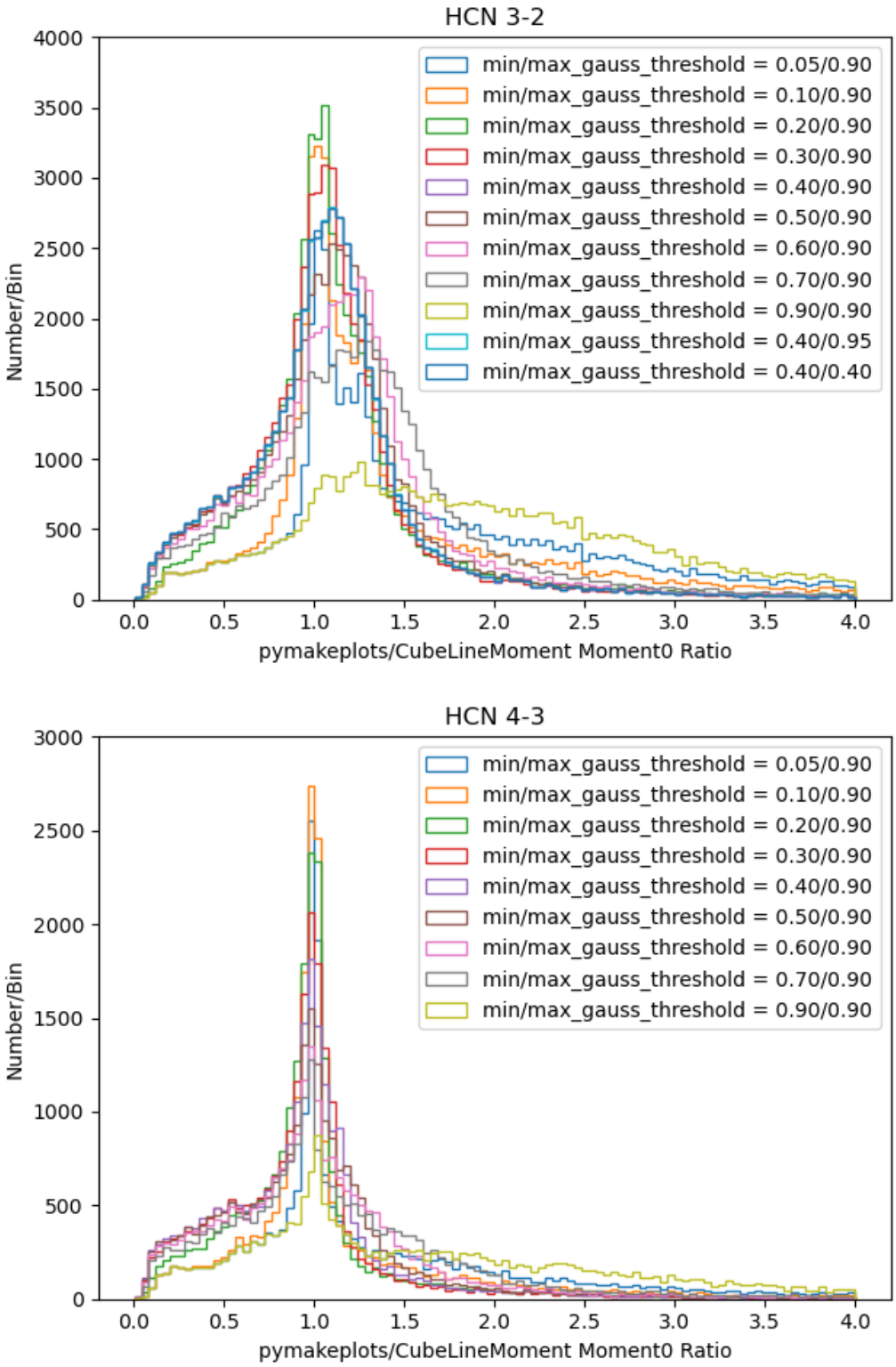

Figure 5: HCN $3-2$ (top) and $4-3$ (bottom) pymakeplots/CubeLineMoment moment 0 ratio pixel histogram distributions for ratios > 0.001. Each histogram represents one of the tests listed in Table 1. Closest agreement between CubeLineMoment and pymakeplots found for min_gauss_threshold equal to 0.3 to 0.4 for both transitions.

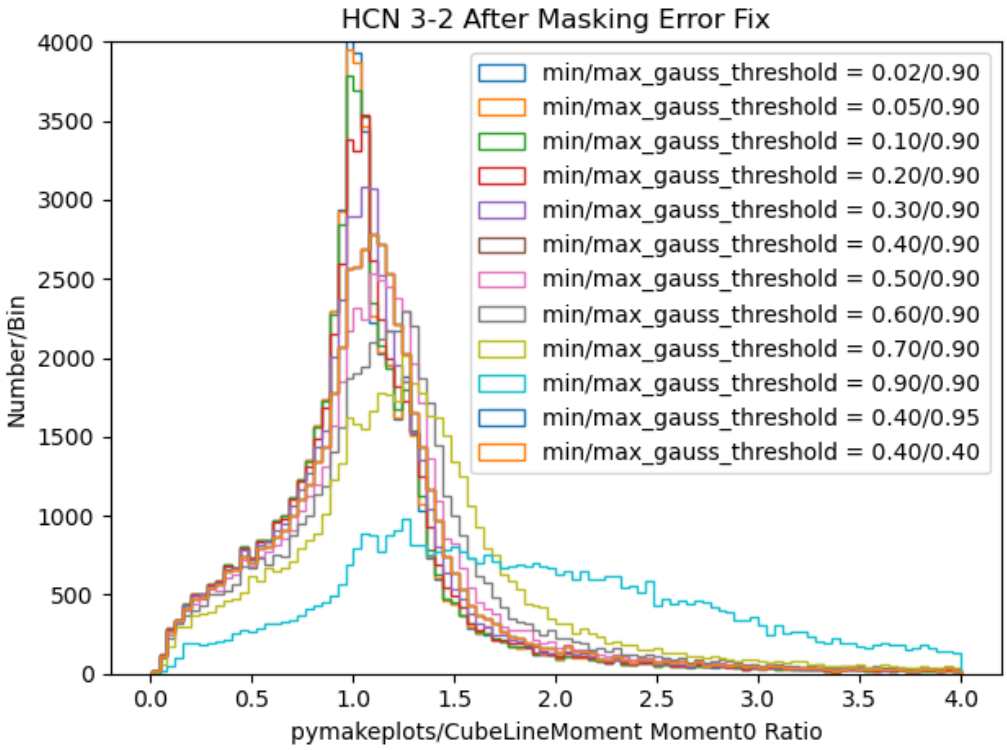

Figure 6: HCN $3-2$ pymakeplots/CubeLineMoment moment 0 ratio pixel histogram distribution after masking error fix (for ratios > 0.001). Each histogram represents one of the tests listed in Table 2. Closest agreement between CubeLineMoment and pymakeplots found for min_gauss_threshold equal to 0.02 to 0.10.

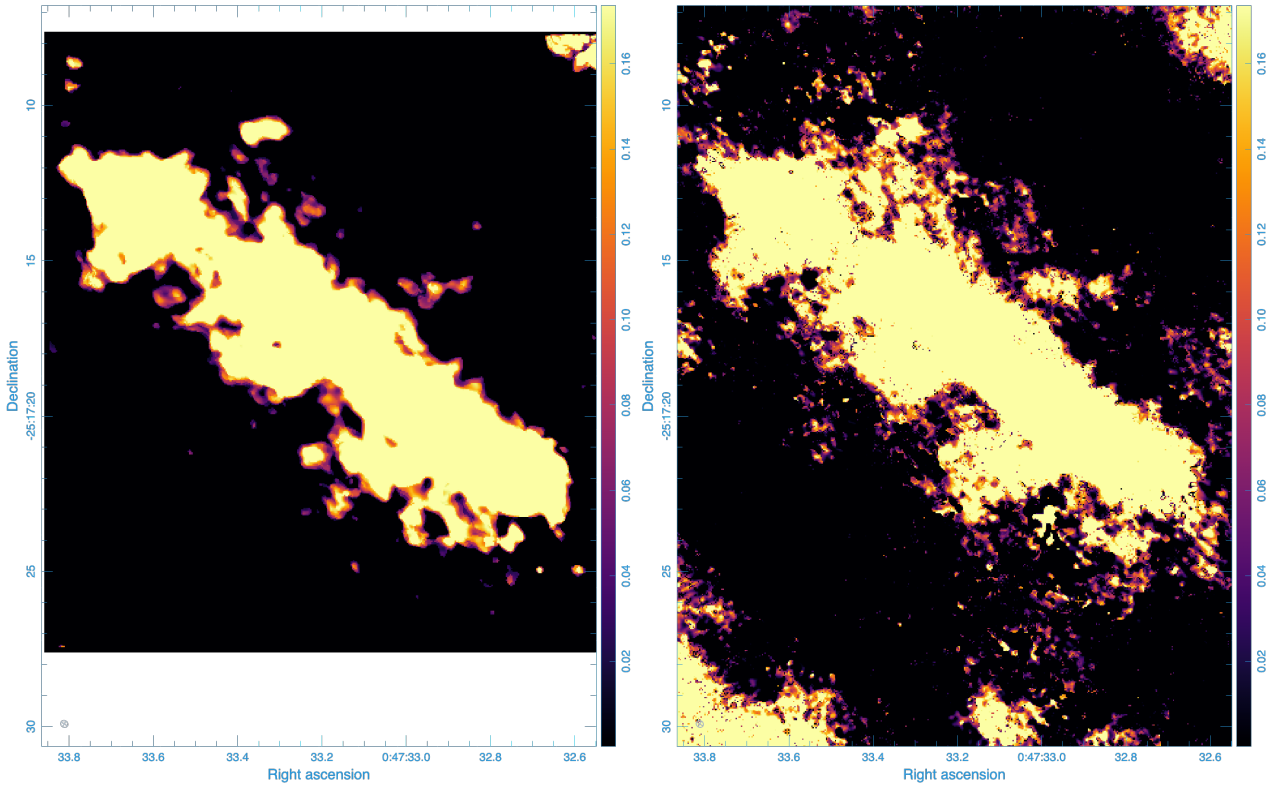

Figure 7: Updated zoomed version of the HCN $3-2$ moment 0 images derived from (left) pymakeplots and (right) CubeLineMoment (after masking fix, using min/max_gauss_threshhod = 0.10/0.90). Note that we are showing more of the area around then CubeLineMoment moment0 image to emphasize the fact that CubeLineMoment includes noise when pymakeplots does not.

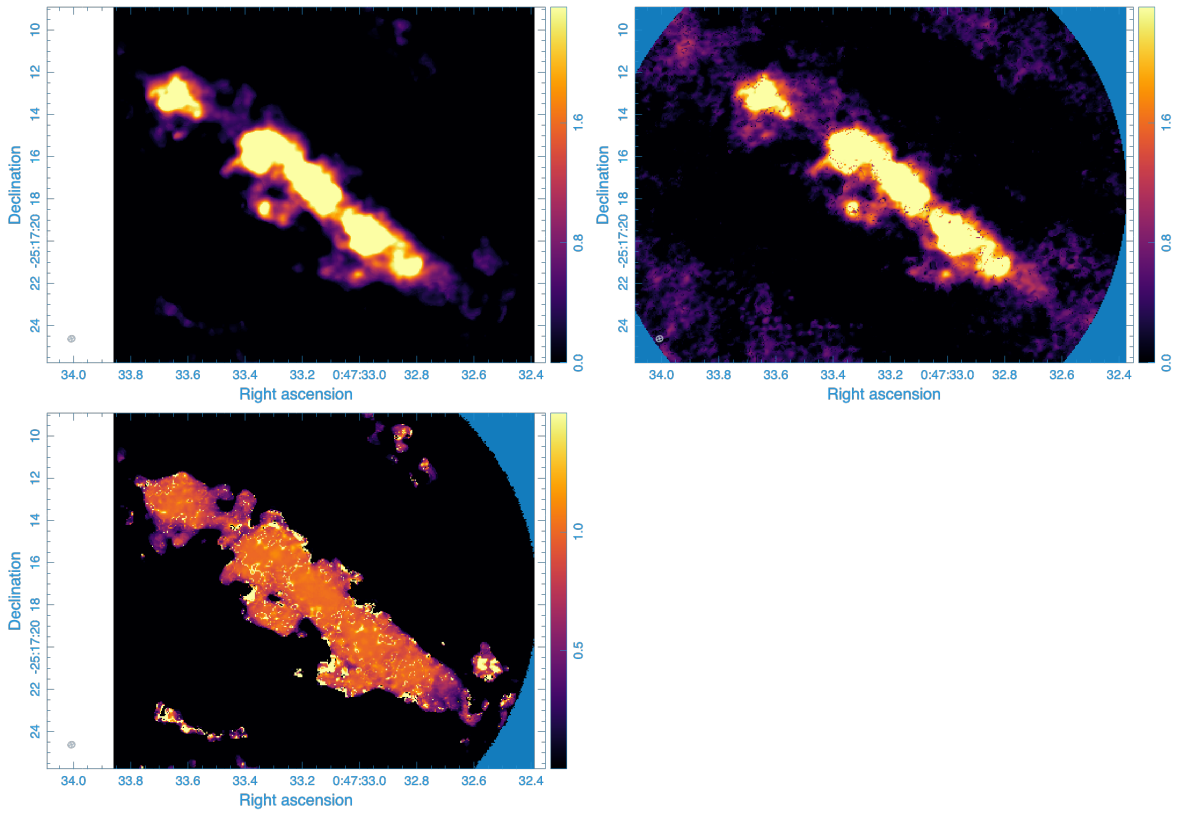

Figure 8: Updated version of the HCN $4-3$ moment 0 images derived from (left) pymakeplots and (right) CubeLineMoment (after masking fix, using min/max_gauss_threshhod = 0.10/0.90, but with no dilation), and (bottom) the ratio of the pymakeplots and CubeLineMoment images. Moment0 images are on the same spatial and intensity scale, and all three images are on the same spatial scale.

Table 1: pymakeplots/CubeLineMoment Dependence on Gaussian Threshold Parameters

min/max_gauss_threshold[a]	$\frac{np.sum(ma)}{len(ma)}$ [b]	DB1[c]	DB2[c]	DB3[c]	DB4[c,d]
HCN $3-2$					
0.05/0.90	4.85	4894938	133504/613089	2134/481719	74/783
0.10/0.90	4.40	5176726	133499/613089	2129/481719	74/783
0.20/0.90	2.45	6602878	133463/613089	2093/481719	74/783
0.30/0.90	1.66	8935802	133441/613089	2071/481719	74/783
0.40/0.90	1.65	10367834	133424/613089	2054/481719	74/783
0.50/0.90	1.71	10502748	133409/613089	2039/481719	74/783
0.60/0.90	1.81	9661269	133398/613089	2028/481719	74/783
0.70/0.90	2.24	8325453	133393/613089	2023/481719	74/783
0.90/0.90	5.19	4651042	133512/613089	2142/481719	74/783
0.40/0.95	1.65	9873110	133663/613089	2293/481719	74/783
0.40/0.40	1.64	13542083	133390/613089	2020/481719	74/783
HCN $4-3$					
0.05/0.90	3.37	1595789	42538/199809	62/157333	61/447
0.10/0.90	2.95	1695476	42535/199809	59/157333	61/447
0.20/0.90	1.84	1992433	42531/199809	55/157333	61/447
0.30/0.90	1.14	2598969	42525/199809	49/157333	61/447
0.40/0.90	1.12	3135309	42512/199809	36/157333	61/447
0.50/0.90	1.19	3273282	42504/199809	28/157333	61/447
0.60/0.90	1.28	3085217	42496/199809	20/157333	61/447
0.70/0.90	1.54	2686839	42491/199809	15/157333	61/447
0.90/0.90	3.77	1500815	42545/199809	69/157333	61/447

[a] Default CubeLineMoment values shown in **bold**
[b] ma = pymakeplots/CubeLineMoment pixel distribution for ratios > 0.001
[c] Debug (DB) Columns: Values Above Thresh, Spatial Pixels Excl,
Spatial Pixels Excl in Region, Mask Cube/Threshold Shapes
[d] For DB4: nn/mmm $\equiv$ mask cube = (nn,mmm,mmm) threshold = (mmm,mmm)
Min/Max always 0.0/1.0

This dilation option seems to do a good job of filling-in the pixel-sized holes in the CubeLineMoment moment0 image very well (Figure 9). It also improved the weighted sum of the ratio of pymakeplots/CubeLineMoment from 1.52 to 1.37. I also repeated the pymakeplots and CubeLineMoment comparison, with dilation, for HCN $4-3$ (Figure 10). The weighted sum has remained unchanged between the un-dilated and dilated weighted sums at 1.03.

5 Conclusions

For both HCN $3-2$ and $4-3$ the masked-moment moment0 (pymakeplots) image retains significantly more of the extended emission than the CubeLineMoment moment0 image. A summary of this comparison is as follows for pymakeplots/CubeLineMoment with min/max_gauss_threshold = 0.10/0.90:

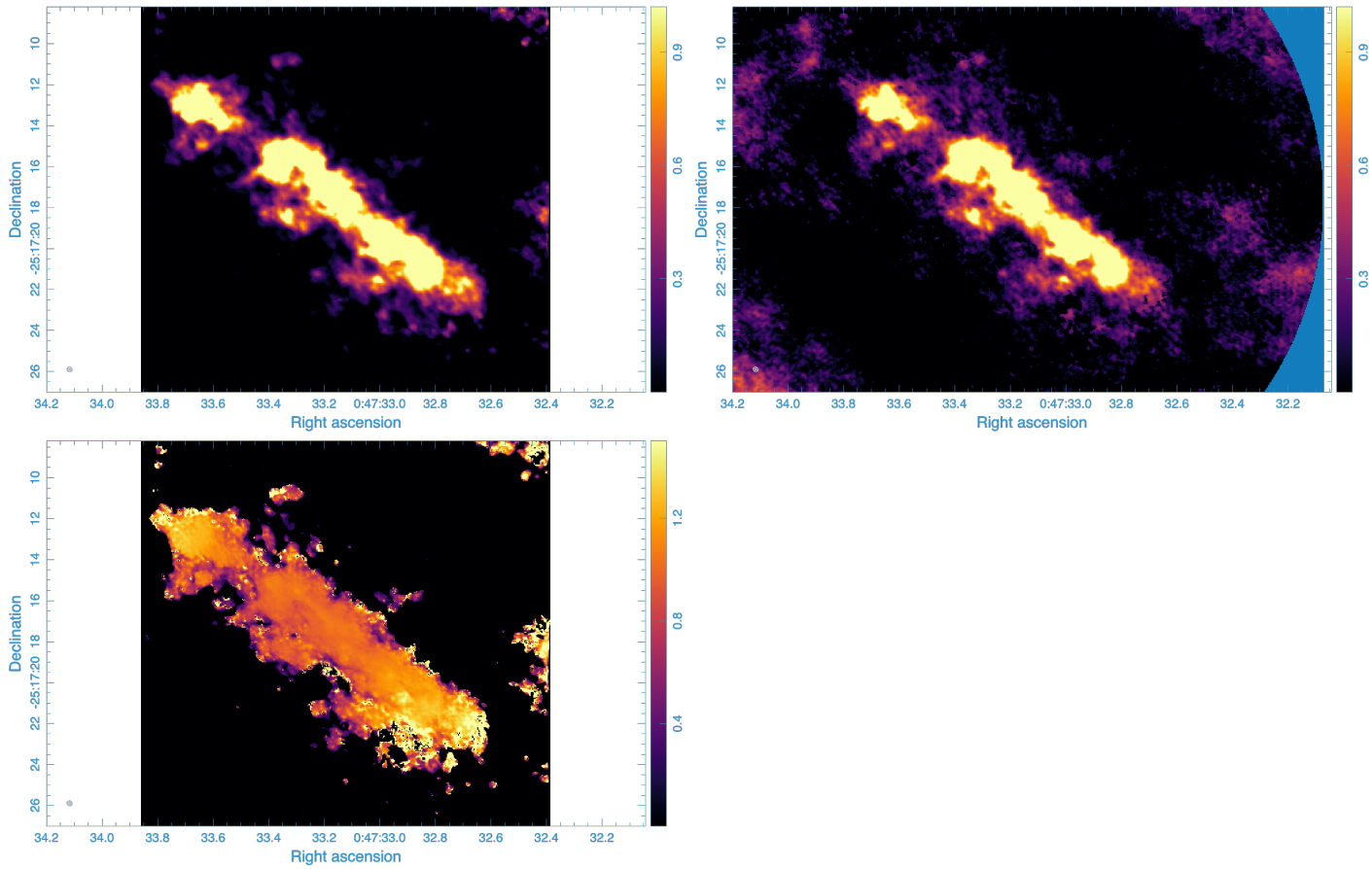

Figure 9: Updated zoomed version of the HCN $3-2$ moment 0 images derived from (left) pymakeplots and (right) CubeLineMoment (after masking fix, using min/max_gauss_threshhod = 0.10/0.90 and including dilation), and (bottom) the ratio of the pymakeplots and CubeLineMoment images. Moment0 images are on the same spatial and intensity scale, and all three images are on the same spatial scale.

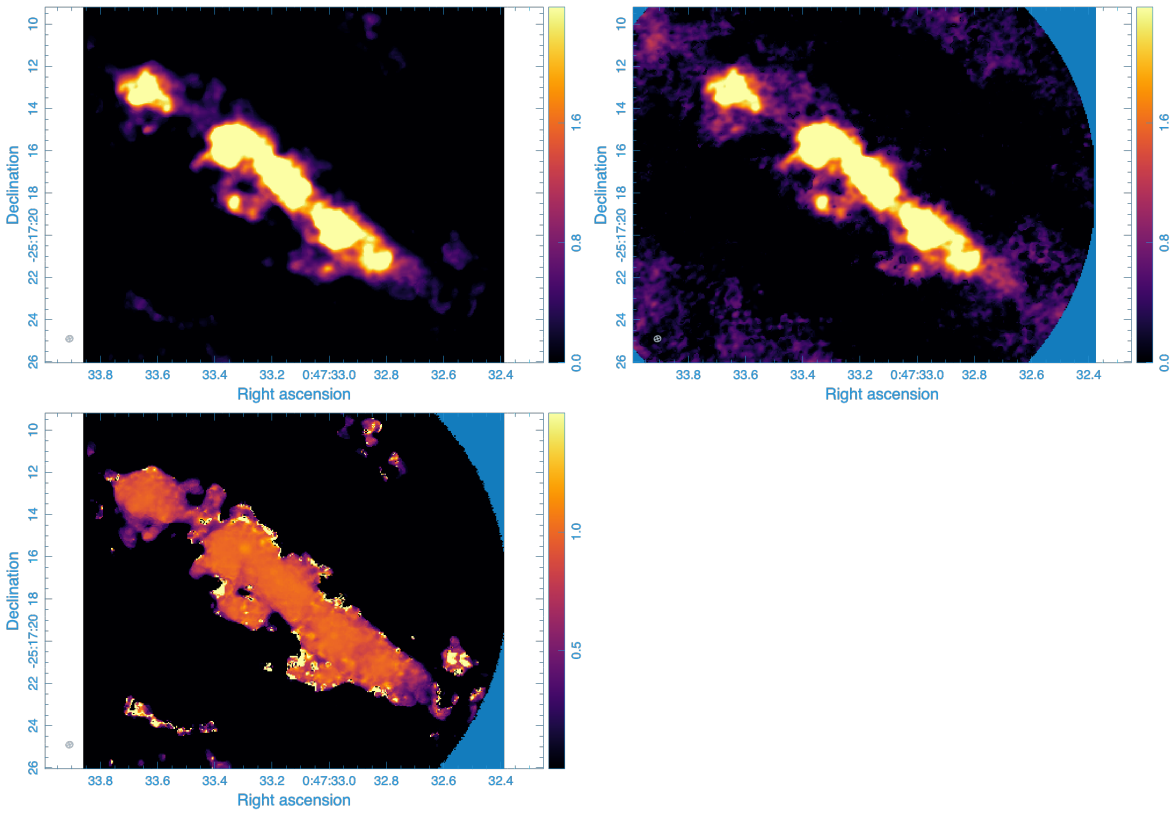

Figure 10: Updated zoomed version of the HCN $4-3$ moment 0 images derived from (left) pymakeplots and (right) CubeLineMoment (after masking fix, using min/max_gauss_threshhod = 0.10/0.90 and including dilation), and (bottom) the ratio of the pymakeplots and CubeLineMoment images. Moment0 images are on the same spatial and intensity scale, and all three images are on the same spatial scale.

Table 2: pymakeplots/CubeLineMoment Dependence on Gaussian Threshold Parameters **After Fix**

min/max_gauss_threshold[a]	$\frac{np.sum(ma)}{len(ma)}$[b]	DB1[c]	DB2[c]	DB3[c]	DB4[c,d]
HCN $3-2$					
0.02/0.90	1.52	14585217	133391/613089	2021/481719	74/783
0.05/0.90	1.52	14562879	133391/613089	2021/481719	74/783
0.10/0.90	1.52	14501726	133391/613089	2021/481719	74/783
0.20/0.90	1.55	14260554	133391/613089	2021/481719	74/783
0.30/0.90	1.58	13648797	133391/613089	2021/481719	74/783
0.40/0.90	1.64	12642774	133391/613089	2021/481719	74/783
0.50/0.90	1.71	10502748	133409/613089	2039/481719	74/783
0.60/0.90	1.81	9955697	133391/613089	2021/481719	74/783
0.70/0.90	2.24	8407776	133391/613089	2021/481719	74/783
0.90/0.90	5.19	4651042	133512/613089	2142/481719	74/783
0.40/0.95	1.64	12623871	133399/613089	2229/481719	74/783
0.40/0.40	1.64	13542083	133390/613089	2020/481719	74/783
HCN $4-3$					
0.10/0.90	1.03	...	...	...	...

[a] Default CubeLineMoment values shown in **bold**
[b] ma = pymakeplots/CubeLineMoment pixel distribution for ratios > 0.001
[c] Debug (DB) Columns: Values Above Thresh, Spatial Pixels Excl,
Spatial Pixels Excl in Region, Mask Cube/Threshold Shapes
[d] For DB4: nn/mmm $\equiv$ mask cube = (nn,mmm,mmm) threshold = (mmm,mmm)
Min/Max always 0.0/1.0

- $\frac{np.sum(ma)}{len(ma)} = 4.40$ for HCN $3-2$ before CubeLineMoment masking bug fix.
- $\frac{np.sum(ma)}{len(ma)} = 2.95$ for HCN $4-3$ before CubeLineMoment masking bug fix.
- $\frac{np.sum(ma)}{len(ma)} = 1.52$ for HCN $3-2$ after CubeLineMoment masking bug fix.
- $\frac{np.sum(ma)}{len(ma)} = 1.03$ for HCN $4-3$ before CubeLineMoment masking bug fix.
- Including dilation in CubeLineMoment results in more pleasing-looking moment0 images.
- Dilation improved the $\frac{np.sum(ma)}{len(ma)}$ from 1.52 to 1.37 for HCN $3-2$ but did not change the weighted sum for HCN $4-3$ (1.03). See Figure 11 for the final ratio distribution histograms for both HCN $3-2$ and $4-3$.
- At least in these two tests, it appears that pymakeplots leaves more extended emission in moment0 calculations than CubeLineMoment. What is not clear is which is a better representation of the integrated emission from a spectral line.

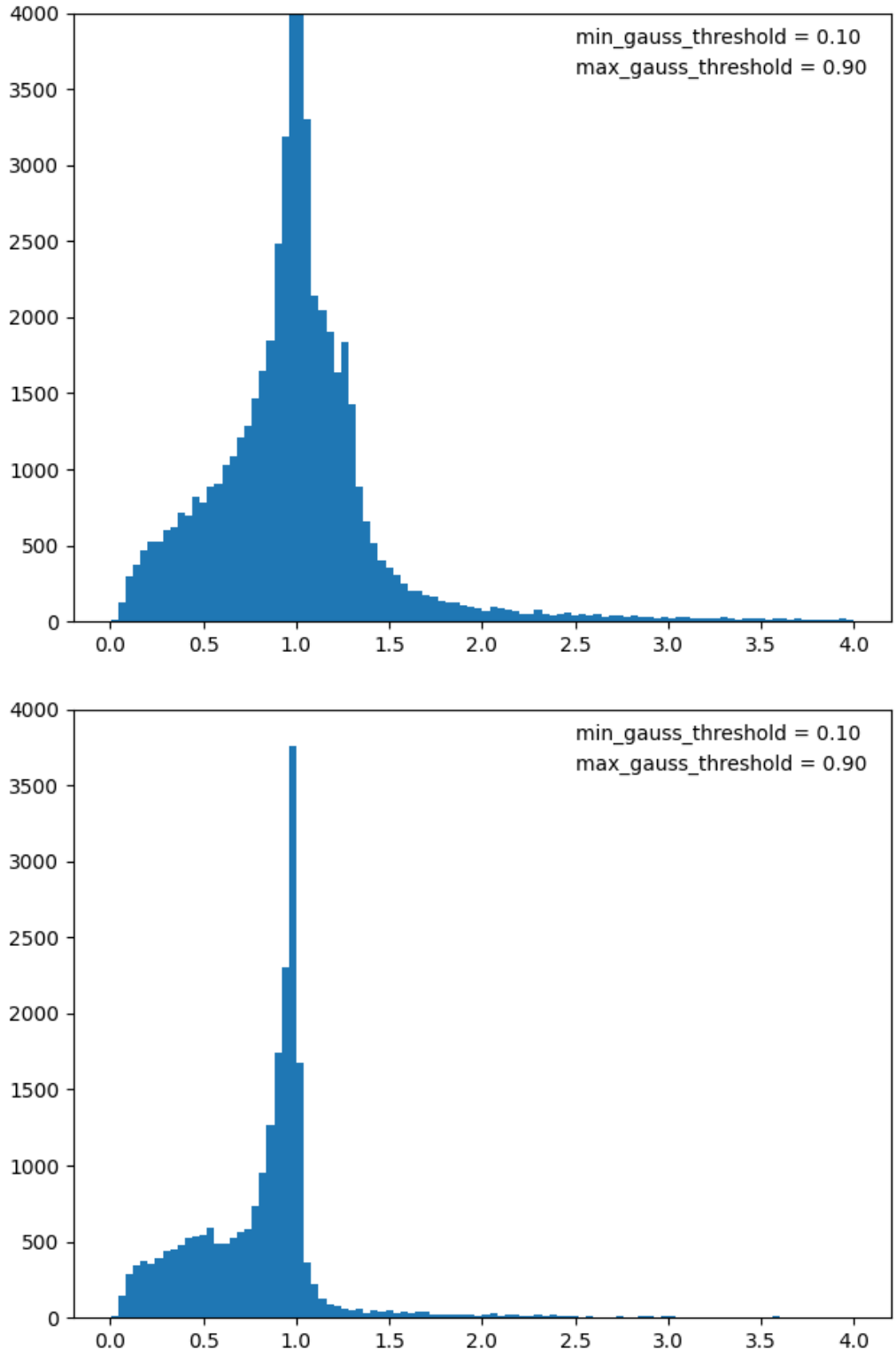

Figure 11: HCN $3-2$ (top) and $4-3$ (bottom) pymakeplots/CubeLineMoment moment 0 ratio pixel histogram distributions for ratios > 0.001. These histograms represent the best alignment between pymakeplots and CubeLineMoment for moment0 calculations.

References

Dame, T. M. 2011, arXiv e-prints, arXiv:1101.1499, doi: 10.48550/arXiv.1101.1499

Rots, A. H., Bosma, A., van der Hulst, J. M., Athanassoula, E., & Crane, P. C. 1990, AJ, 100, 387, doi: 10.1086/115522