Lupus 3 mms outflow observed with ALMA

Adele Plunkett (National Radio Astronomy Observatory, Charlottesville, VA, USA)
Luke Maud (ESO Garching)
Fanyi Meng (Uni-Cologne)
Teresa Orozco-Aguilera (INAOE, Puebla, México)
María Mercedes Vazzano (IAR, Argentine)
Veena V. S (I. Physikalisches Institut, Universität zu Köln, Kön, Germany)
Per Bjerkeli (Chalmers)
Aida Ahmadi (Max Planck Institute for Astronomy, Heidelberg, Germany)
Yo-Ling Chuang (National Taiwan Normal University, Taipei, Taiwan)
Yi-Jehng Kuan (National Taiwan Normal University, Taipei, Taiwan)
September 4, 2019

This memo was prepared as part of the workshop "Improving Image Fidelity on Astronomical Data: Radio Interferometer and Single-Dish Data Combination," held on 12-16 Aug 2019 at the Lorentz Center in Leiden, The Netherlands.

1 Dataset overview

The dataset used for this memo is that of ALMA#2015.1.00306.S (PI: Adele Plunkett). The observations are of ^{12}CO (1-0) that probes the molecular outflow from the protostar Lupus 3 mms. Several important observing parameters are given in Table 1.

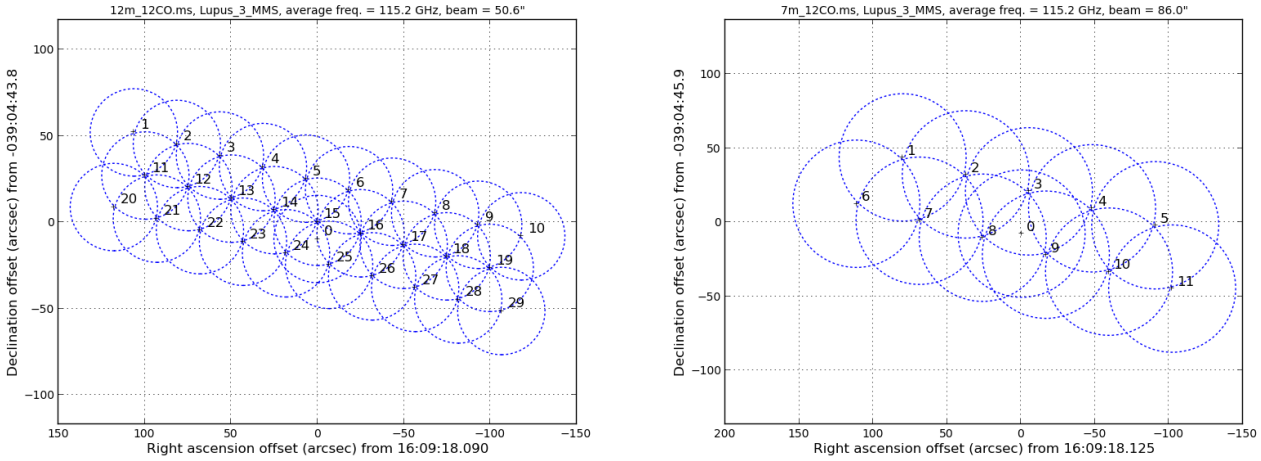
Table 1: Observational details

Parameter	Value
Phase center	J2000 16h09m18.1 -39d04m44.0
Rest frequency	115.27120GHz
V_{LSR}	4 km/s
ΔV (channel width)	1 km/s
Velocity range imaged	[5-8] km/s
Map size	$300'' \times 120''$
12m-array pointings	29
7m-array pointings	11
12m-array beamsize	$1\farcs7 \times 1\farcs4$ (PA=72°2)
7m-array beamsize	$14\farcs9 \times 8\farcs6$ (PA=84°6)
Range of 12m baselines	[15 - 453] m
Range of 7m baselines	[9 - 49] m

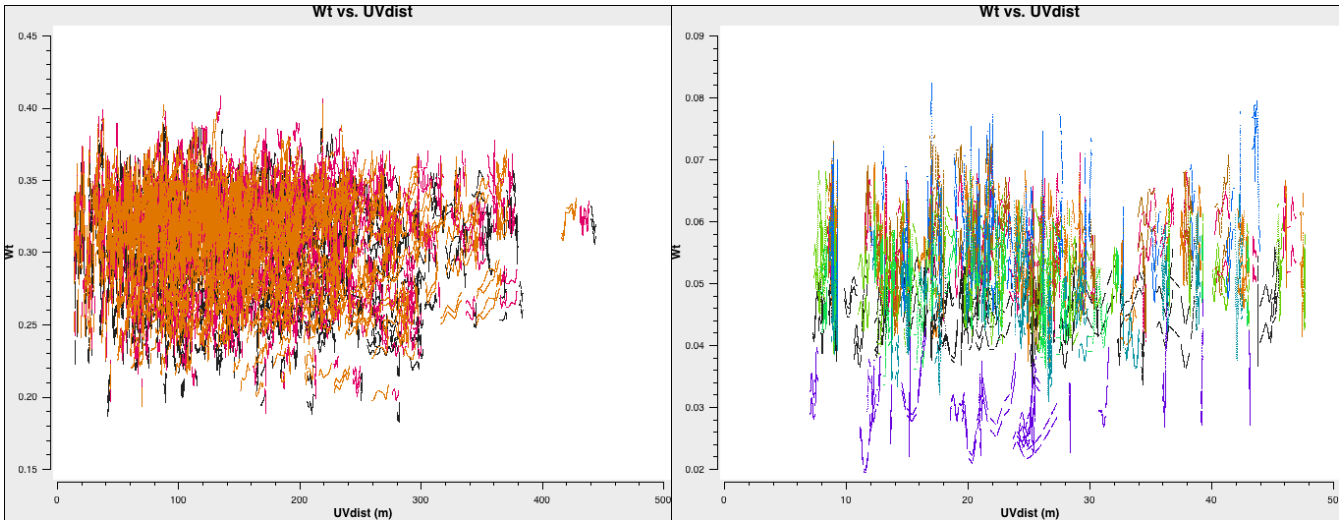
2 Combination Methods

2.1 Method 1: Feather

The feather method follows that of the CasaGuide `https://casaguides.nrao.edu/index.php/M100_Band3_Combine_5.4`.



(a) Mosaics (12m on left; 7m on right)



(b) Weights (12m on left; 7m on right). Notice that one SPW in the 7m data seems to have lower weightings, so be safe we can omit this SPW.)

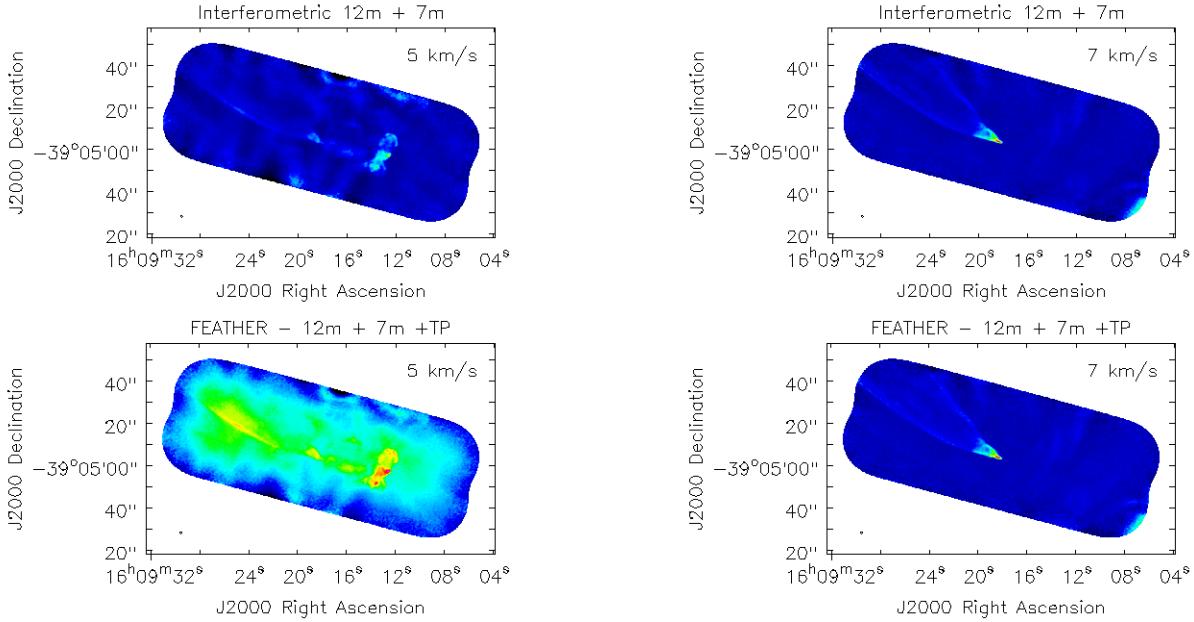
Figure 1: Preliminary data assessment.

LM: Running CASA 5.4.0-70, MAC OS10.14: In order to use feather, as per the above guide, the TP fits image was re-gridded, the reference frequency reset and then the 12m+7m primary beam was multiplied in. Subsequently, feather was used without extra options in order to produce the merged image. One of the major difficulties, especially with a dataset such as the Lupus Outflow are the vast range of structures and artifacts in the interferometric image. This requires very careful cleaning before the feather task, otherwise feather will not 'fix' the imprinted imaging artifacts due to the interferometric sampling.

2.2 Method 2: Joint Deconvolution (tp2vis)

We succeeded in using the tp2vis method, but revealed the challenge when strong emission extends to/beyond the edge of the single dish map. In figure **??** we show a few moment maps.

The important parameter in this case (with strong emission near the edge) is the *winpix* in the tp2vis command. We use the following :



(a) Complex Emission Channel at 5km/s

(b) Line wing Emission Channel

Figure 2: Feather image in complex and simple structure channels (5 and 7km/s respectively), Interferometric: Top, Feathered: Bottom. LM: CASA 5.4.0-70, MAC OS 10.14.

```
tp2vis(TPim,'tp_winpx9.ms','12.ptg',rms=TPrms,winpix=9)
```

We tested the default winpix (=0); as well as the values 3, 6, 9. The winpix parameter indicates the Tukey window for taper, which is known to work well with Fourier Transform. The pixels are based on the single dish image, and blanks around the edge of an image are ignored.

After creating the pseudo-visibilities, a simple task `tp2vispl` allows us to visualize the uv-coverage, and assess the amplitudes and weights, especially important in the overlap region:

```
tp2vispl(['12m.ms','7m.ms','tp.ms'],outfig="tp2vispl.png")
```

LM: CASA 5.4.0-70 MAC OS10.14: TP2VIS was not successful, although the mechanics functioned, the TP image input could not be recovered correctly. Most notable was that the linear structure of the outflow was not positioned correctly and appeared at the edges of the map. Testing the TP2VIS with and without the deconvolve step indicated that there was an issue with this stage - either due to the casa version or the operating system. CASA 5.6.0-60 was trialed with better results. However, the PSF resulting for the TP from clean was not a simple Gaussian, but an oval shape with a clear sidelobe. Merging was not yet undertaken.

2.3 SDINT method

Method developed by Rau et al. (2019). Combine SD and INT images and PSFs via feathering prior to minor cycle deconvolution, but keep the data (and major cycles) separate. To apply this method we use the PSF previously obtained by the tp2vis method.

After setting the parameters we ran the script[1]

[1]https://github.com/urvashirau/WidebandSDINT

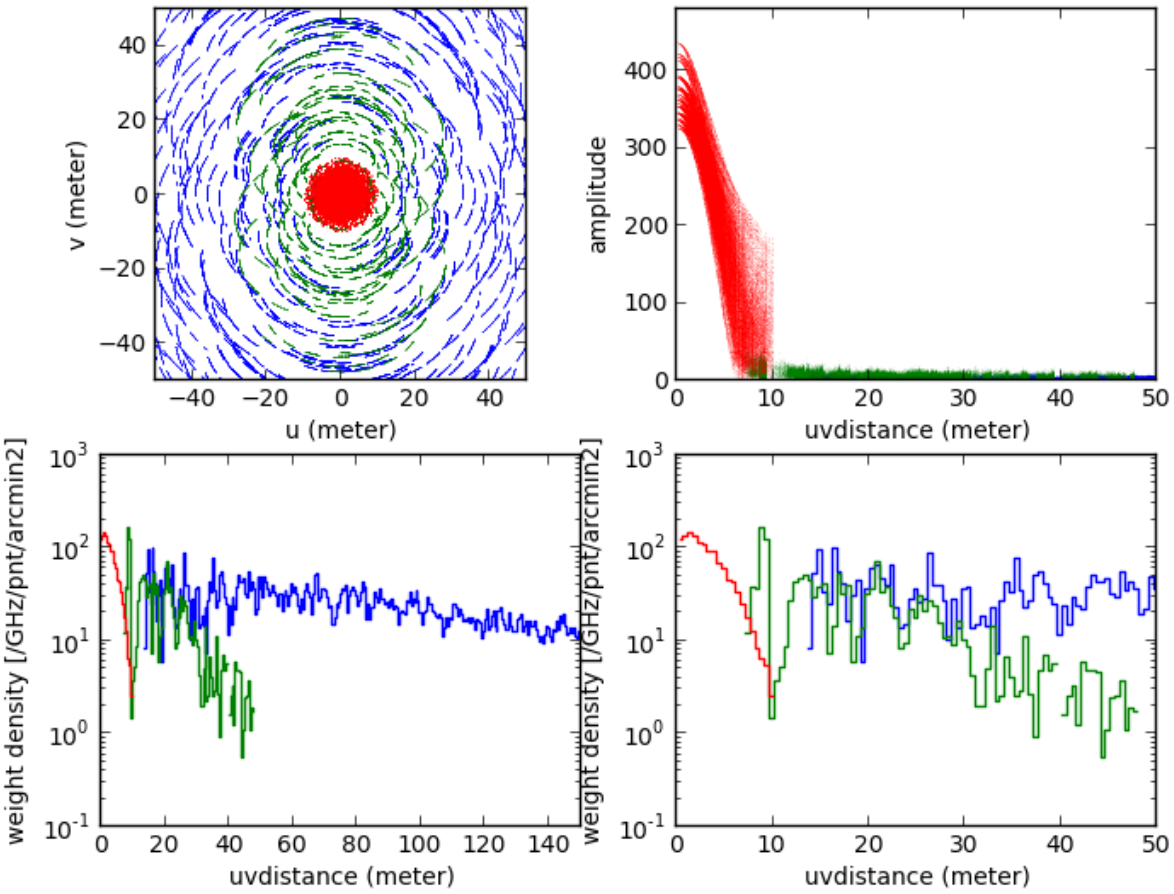


Figure 3: Output of tp2vispl for the 12m, 7m, and total power visibilities. Red indicates TP; green 7-m array; and blue 12-m array.

```
execfile('runsdint.py')
```

3 Comparison and evaluation

In this section, you present the thorough (and challenging) work to both qualitatively **and quantitatively** compare the methods, and evaluate which one works better/best in the case of the dataset you tested. **These methods are not directly provided to you. Instead, you should develop in your group the methods to evaluate the "goodness" of each combination.**

Be as specific as possible. If you know *why* one method worked differently than another, if you can include equations, snippets of code (here, or in Appendix below), or plots of quantitative metrics, then we can deeply understand and tackle the sometimes subtle differences in the methods. Ideally, these code snippets will be useful to users in the future when they publish combined data, in order to assess the outcome.

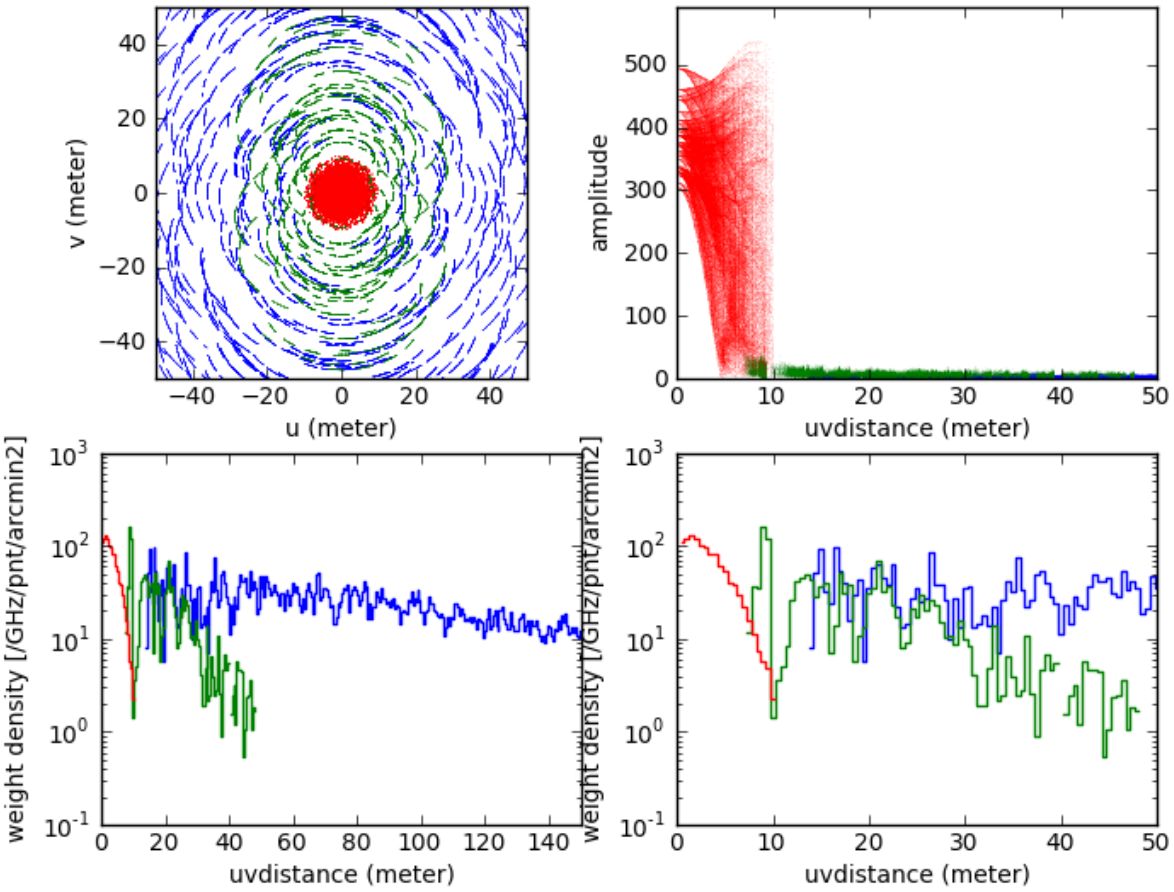


Figure 4: Output of tp2vispl for the 12m, 7m, and total power visibilities. Red indicates TP; green 7-m array; and blue 12-m array (Veena V. S).

4 Additional notes, comments, challenges encountered

Before applying any combination method it is necessary to regrid the TP image to match the shape of the 7m+12m image using the CASA task `imregrid`. It is also necessary to check that the order of the axes of cubes (TP and 7+12m) coincide. Otherwise it is possible to transpose one of the cubes with the CASA task `imtarns`.

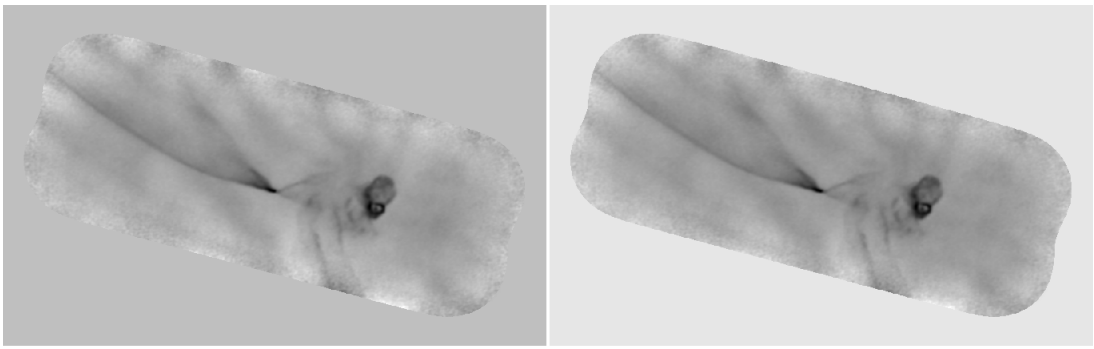


Figure 5: Combined image (SDINT method) using as TP input the cube obtained from the pseudovisibilities obtained with tp2vis method (left) and the TP cube (right).

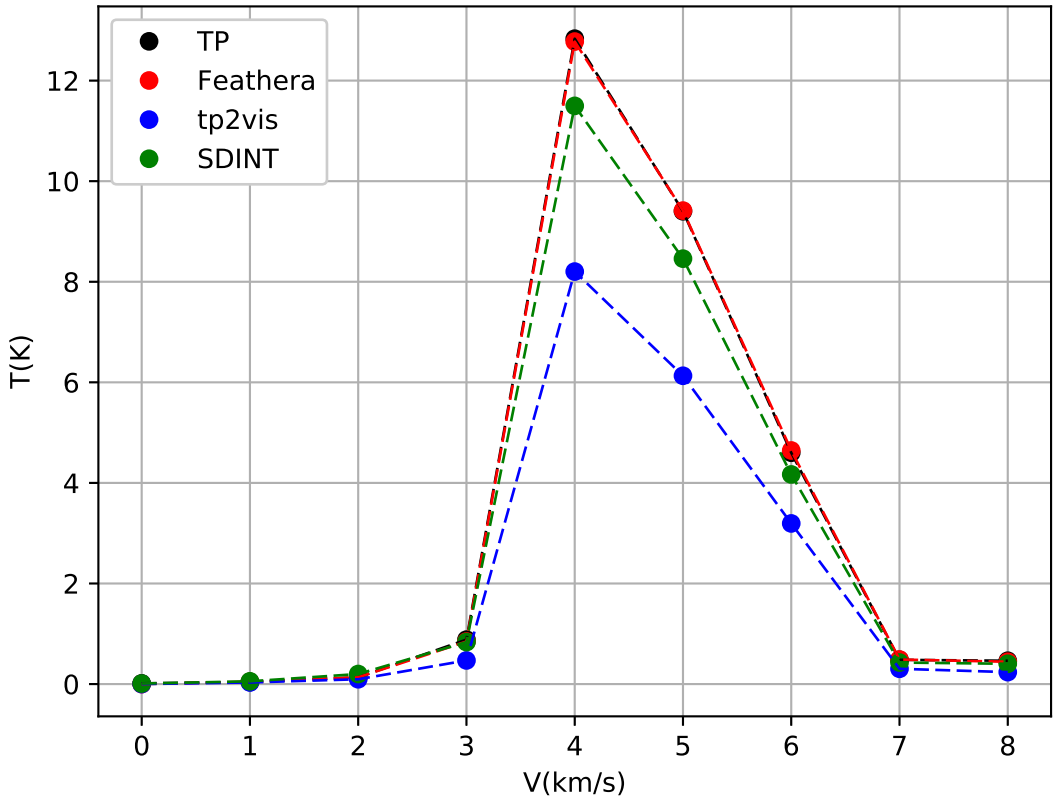


Figure 6: Comparison of spectra obtained from data cubes combined with different methods and TP cube. Feather is the only method that recovers the whole emission.

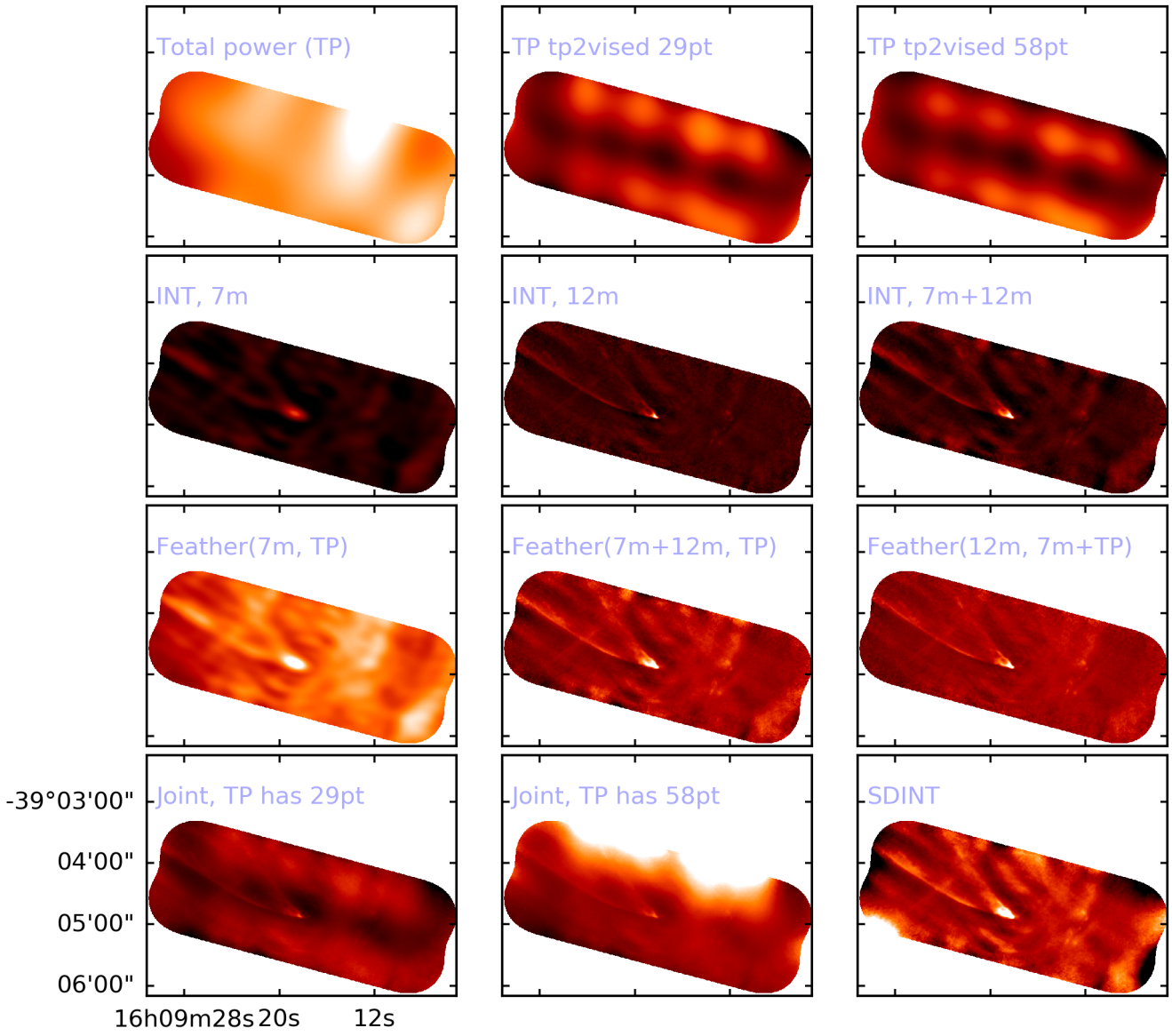


Figure 7: Channel 5 km/s, different methods, DIFFERENT upper and lower limits, all with linear scale. CASA V5.4 (Fanyi Meng)

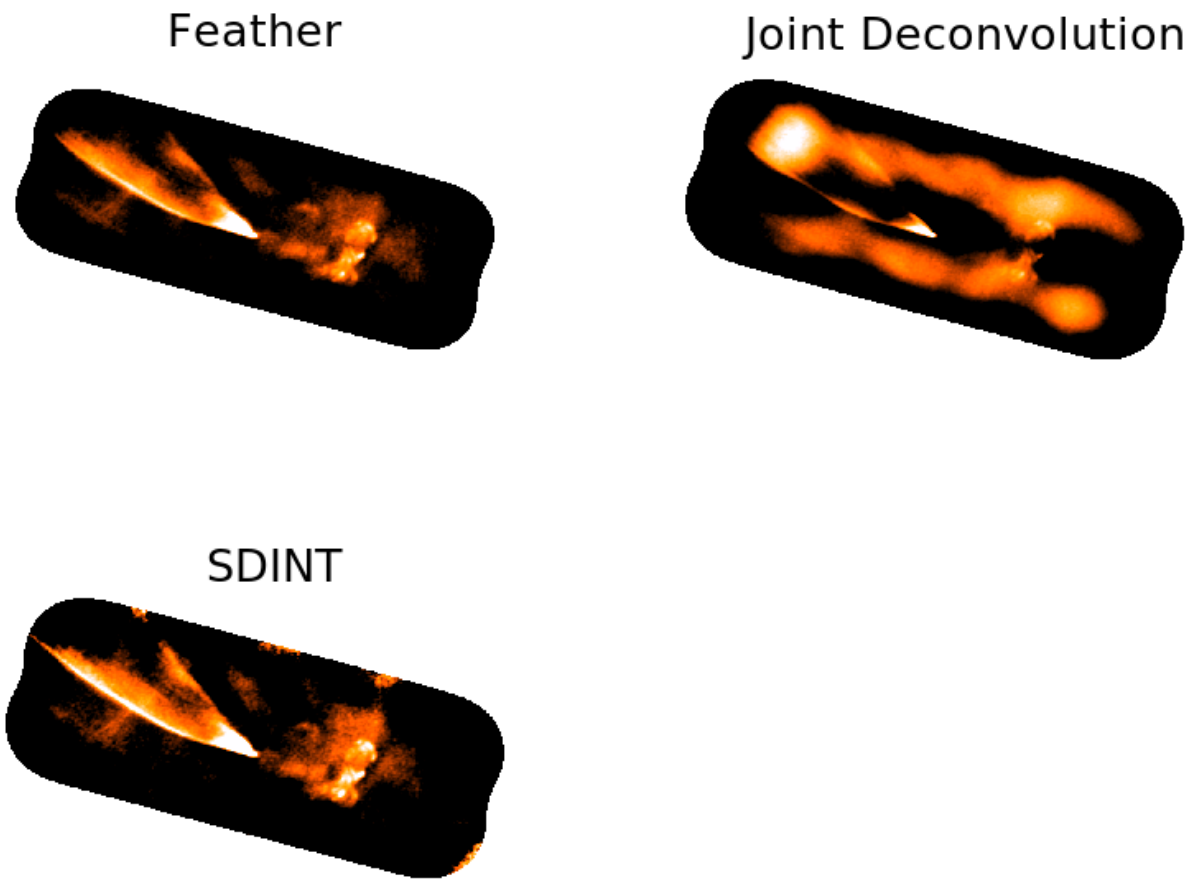


Figure 8: Integrated intensity map in the velocity range 1-5 km/s using different methods (Veena V. S). CASA V5.6

5 Appendix: Feather Code

```
################################
## LUPUS 3 Outflow Working Group
## Started by Adele Plunkett, 14 August 2019
## Will continue to update during/after the workshop
################################



'''
NOTE: The interferometry data were cut down to 10 channels of 1 km/s each by the
    ↪ following commands:
line = {âĂrestfreqâĂİ : âĂŸ115.27120GHzâĂŹ,
 âĂŸstartâĂŹ : âĂŸ-1.0km/sâĂŹ,
 âĂŸwidthâĂŹ : âĂŸ1.0km/sâĂŹ,
 âĂŸnchanâĂŹ : 10,}
mstransform(sharedir+vis7m,âĂŸmst_07_nchan10_start0kms.msâĂŹ,
 datacolumn=âĂŸDATAâĂŹ,outframe=âĂŸLSRKâĂŹ,mode=âĂŸvelocityâĂŹ,regridms=True,
    ↪ keepflags=False,
 **line)
mstransform(sharedir+vis12m,âĂŸmst_12_nchan10_start0kms.msâĂŹ,
 datacolumn=âĂŸDATAâĂŹ,outframe=âĂŸLSRKâĂŹ,mode=âĂŸvelocityâĂŹ,regridms=True,
    ↪ keepflags=False,
 **line)
'''

## Data can be acquired here (for now...): ftp://ftp.astro.umd.edu/pub/teuben/
    ↪ tp2vis/Lup3mms_12CO_tp_7m_12m_nchan10.tgz
datadir = 'path/DataComb2019/Lup3mms/Lup3mms_Share/' #the directory where your
    ↪ test data are held
vis7m = datadir+'mst_07_nchan10_start0kms.ms'
vis12m = datadir+'mst_12_nchan10_start0kms.ms'
TPfits = datadir+'TP_12CO.fits'
TPim = 'TP.image' #You choose the name of your TP image
#importfits(fitsimage=TPfits,imagename=TPim) ## You just need to do this the
    ↪ first time, in order to get your TP Fits file into the CASA format (*.image
    ↪ )
vis12m7m = 'int12m7m.ms' #You choose the name of your combined interferometry
    ↪ image


###############
## Day 1: FEATHER
## Generally, follow the CasaGuide https://casaguides.nrao.edu/index.php/
    ↪ M100_Band3_Combine_5.4
###############
```

```
# In CASA, look at mosaic map pointings
os.system('rm -rf *m_mosaic.png')
iminfo12m = au.plotmosaic(vis12m,sourceid='0',figfile='12m_mosaic.png')
iminfo7m = au.plotmosaic(vis7m,sourceid='0',figfile='7m_mosaic.png')
## I also save the output in the variables iminfo, because they hold the size of
   ↪ the image we will want to create, in arcsec.
imsize_x = iminfo7m[1]-iminfo7m[2]
imsize_y = iminfo7m[3]-iminfo7m[4]
print('# Image size should be approximately: {0:.0f} x {1:.0f} arcsec'.format(
   ↪ imsize_x,imsize_y))
# Image size should be approximately: 392 x 267 arcsec

# Look at how many spectral windows you have in each *.ms
listobs(vis=vis7m,listfile='7m.listobs')
listobs(vis=vis12m,listfile='12m.listobs')
# In this case, 9 spws with 7m; 3 spws with 12m

# Look at weights
# In CASA
os.system('rm -rf 7m_WT.png 12m_WT.png')
plotms(vis=vis12m,yaxis='wt',xaxis='uvdist',spw='0~2:200',
      coloraxis='spw',plotfile='12m_WT.png')
#
plotms(vis=vis7m,yaxis='wt',xaxis='uvdist',spw='0~8:200',
      coloraxis='spw',plotfile='7m_WT.png')

# We found that the SPW 7 has lower amplitudes, therefore we choose to remove
   ↪ that in this case
split(vis=vis7m,outputvis='int7m.ms.spl',spw='0,1,2,3,4,5,6,8',datacolumn='data')
vis7m = 'int7m.ms.spl'

# Concat, you could also scale weights, but the weights here look good already
os.system('rm -rf *12m7m.ms')
concat(vis=[vis12m,vis7m],concatvis=vis12m7m)
listobs(vis=vis12m7m,listfile='12m7m.listobs')

#concat(vis=[vis12m,vis7m],concatvis=vis12m7m+'.cpfalse', copypointing=False)

## Concat may give a warning like this, and I ignored it:
#2019-08-17 14:26:05 WARN MSConcat::copySysCal /Users/aplunket/ResearchNRAO/
   ↪ Meetings/DataComb2019/Lup3mms/Post-workshop/12m7m.ms does not have a valid
   ↪ syscal table,
#2019-08-17 14:26:05 WARN MSConcat::copySysCal+ the MS to be appended, however,
   ↪ has one. Result won't have one.
#2019-08-17 14:26:05 WARN MSConcat::concatenate (file ../../ms/MSOper/MSConcat.cc
   ↪ , line 825) Could not merge SysCal subtables
```

```
####################################
## CLEAN the interferometry data
####################################

'''
Use these *.ms:
datadir = 'path/DataComb2019/Lup3mms/Lup3mms_Share/' #the directory where your
    ↪ test data are held
vis12m = datadir+'mst_12_nchan10_start0kms.ms'
vis12m7m = '12m7m.ms' #You choose the name of your combined interferometry image
vis7m = '7m.ms.spl'

'''

field='Lupus_3_MMS*' # Look in the log to find the name of your source
phasecenter='J2000 16h09m18.1 -39d04m44.0' # Look in the listobs, or use the known
    ↪  source coordinates

## I have not confirmed this, but I'm told that for efficiency purposes, tclean
    ↪ likes image size numbers factorizable by 2,3,5,7. These are:
'''for i in range(20,5001,5):
     if i%2.==0 and i%3.==0 and i%7.==0:
         print i
'''

## There might be an au task to do this...
expected_hpbw12m = au.estimateSynthesizedBeam(vis12m)
expected_hpbw7m = au.estimateSynthesizedBeam(vis7m)
print('# Expected beamsize for 12m: {0:.2f}; {1:.2f}'.format(expected_hpbw12m,
    ↪ expected_hpbw7m))
print('# Suggest cell for 12m: {0:.1f}; {1:.1f}'.format(expected_hpbw12m/4,
    ↪ expected_hpbw7m/4))
# Expected beamsize for 12m: 1.49; 8.96
# Suggest cell for 12m: 0.4; 2.2
cell12m = 0.4
cell7m = 2.0
##Then calculate image size
imsize12_x_px = imsize_x/cell12m
imsize12_y_px = imsize_y/cell12m
imsize7_x_px = imsize_x/cell7m
imsize7_y_px = imsize_y/cell7m
print('# Suggest imsize (in px) for 12m: [{0:.0f}, {1:.0f}]'.format(imsize12_x_px,
    ↪ imsize12_y_px))
print('# Suggest imsize (in px) for 7m: [{0:.0f}, {1:.0f}]'.format(imsize7_x_px,
    ↪ imsize7_y_px))
# Suggest imsize (in px) for 12m: [981; 668]
# Suggest imsize (in px) for 7m: [196; 134]
```

```
## Note, these are likely too big, so you may round down some.

## 7m parameters
lineimage7m = {"restfreq" : '115.27120GHz',
  'start' : '0.0km/s',
  'width' : '1.0km/s',
  'nchan' : 8,
  'imsize' : [196,160],
  'cell' : '2arcsec',
  'gridder' : 'mosaic',
  'weighting' : 'briggs',
  'robust' : 0.5,
}

## 12m parameters
lineimage12m = {"restfreq" : '115.27120GHz',
  'start' : '0.0km/s', #you can set this in km/s
  'width' : '1.0km/s', #you can set this in km/s
  'nchan' : 8, #you can choose fewer channels to save time; I'm choosing not to
     ↪ image first and last channels
  'imsize' : [896,630], #also tried [896,540]; [900,600]
  'cell' : '0.4arcsec',
  'gridder' : 'mosaic',
  'weighting' : 'briggs',
  'robust' : 0.5,
}

#First we make a "dirty" image of each (niter=0, or no iterations) just to see
   ↪ that the image parameters are OK.
#This test case should not take too long (less than a few minutes)
tclean(vis=vis7m,imagename='int7m_niter0',field=field,spw='',phasecenter=
   ↪ phasecenter,mosweight=True,specmode='cube',niter=0,interpolation='nearest'
   ↪ ,**lineimage7m)

tclean(vis=vis12m,imagename='int12m_niter0',field=field,spw='',phasecenter=
   ↪ phasecenter,mosweight=True,specmode='cube',niter=0,interpolation='nearest'
   ↪ ,**lineimage12m)

tclean(vis=vis12m7m,imagename='int12m7m_niter0',field=field,spw='',phasecenter=
   ↪ phasecenter,mosweight=True,specmode='cube',niter=0,interpolation='nearest'
   ↪ ,**lineimage12m) ##use the parameters for the higher-resolution
   ↪ interferometer when making the combined 12m+7m map

## Compare maps, and make sure they they are covering the same region; that cell
   ↪ size is OK; etc.

#Next, RUN a deeper TCLEAN of each map
niter=50000 ## You can increase this when things are going well.
```

```
cniter=500 ## It was recommended to me to use a smaller (<1000) cniter, but this
    ↪ takes longer
gain=0.05 ## It was recommended to me to use a smaller (<0.1 default) gain, but
    ↪ this takes longer
threshold='0mJy' ## You can clean based on iterations or threshold

## YOUR 12m+7m map
tclean(vis=vis12m7m,imagename='int7m12m_clean',field=field,spw='',phasecenter=
    ↪ phasecenter,mosweight=True,specmode='cube',niter=niter,cycleniter=cniter,
    ↪ gain=gain,threshold=threshold,usemask='pb',pbmask=0.3,**lineimage12m)

## YOUR 12m-only map
tclean(vis=vis12m,imagename='int12m_clean',field=field,spw='',phasecenter=
    ↪ phasecenter,mosweight=True,specmode='cube',niter=niter,cycleniter=cniter,
    ↪ gain=gain,threshold=threshold,usemask='pb',pbmask=0.3,**lineimage12m)

## YOUR 7m-only map
tclean(vis=vis7m,imagename='int7m_clean',field=field,spw='',phasecenter=
    ↪ phasecenter,mosweight=True,specmode='cube',niter=niter,cycleniter=cniter,
    ↪ gain=gain,threshold=threshold,usemask='pb',pbmask=0.3,**lineimage7m)
## I tweaked pbmask to be slightly higher than the default (which is 0.2?)
    ↪ because the edges in this map may be particularly noisy, with bright
    ↪ emission.
## YOU can do more iterations of TCLEAN, until you are happy with the "CLEAN"
    ↪ image

## NOTE: I have run 12m for 20000 iterations each; I could run more.
## NOTE: I have run 7m for 80000 iterations each; I could run more.
## NOTE: I have run 7m12m for 150000 iterations each; Between 100000-150000
    ↪ iterations, a more sophisticated clean might be needed.

imsmooth(imagename='int7m12m_clean.image',outfile='int7m12m.imsmooth',kernel='
    ↪ commonbeam')
imsmooth(imagename='int7m_clean.image',outfile='int7m.imsmooth',kernel='commonbeam
    ↪ ')
imsmooth(imagename='int12m_clean.image',outfile='int12m.imsmooth',kernel='
    ↪ commonbeam') #this is a trick when you need a single beam, rather than a
    ↪ beam per channel. It will be necessary for the feather with TP image. Be
    ↪ careful if you have many channels, as you may be using the incorrect beam.

## SOME warnings liks this, but I ignore it for now: 2019-08-14 07:45:07 WARN
    ↪ imsmooth::Image2DConvolver::_dealWithRestoringBeam Convolving kernel has
    ↪ minor axis 0.0736102 arcsec which is less than the pixel diagonal length of
    ↪  0.565685 arcsec. Thus, the kernel is poorly sampled, and so the output of
    ↪ this application may not be what you expect. You should consider increasing
    ↪  the kernel size or regridding the image to a smaller pixel size
```

```
########################
## Prepare to FEATHER
########################

intimagename = 'int7m12m_clean' #prefix from the TCLEAN command above
#Drop the Stokes axis in both images
imsubimage(imagename='int7m12m.imsmooth',dropdeg=True,outfile='IntForComb.imsmooth
   ↪ .dropdeg')
imsubimage(imagename='TP.image',dropdeg=True,outfile='TPForComb.dropdeg')

Intim = 'IntForComb.imsmooth.dropdeg' #12m+7m image from TCLEAN, after imsubimage
Intpb = intimagename+'.pb' #primary beam response, this was created in your
   ↪ previous 12m+7m TCLEAN
Intmask = intimagename+'.mask' #mask from TCLEAN (cutting below a certain pb
   ↪ value), this was created in your previous 12m+7m TCLEAN
TPim = 'TPForComb.dropdeg'

#Regrid the Inter. mask to have 1 channel
imsubimage(imagename=Intmask,
          chans='0',outfile=Intmask+'.cut',dropdeg=True)
#Regrid the Inter. PB mask to have 1 channel
imsubimage(imagename=Intpb,
          chans='0',outfile=Intpb+'.cut',dropdeg=True)

#check that rest frequencies match - Initially different. You have to use "
   ↪ imreframe" to set the rest frequency of the TP image to match that of 12m
tp_restfreq=imhead(TPim,mode='get',hdkey='restfreq')
int_restfreq=imhead(Intim,mode='get',hdkey='restfreq')
if tp_restfreq == int_restfreq: print('##␣Frequencies␣match')
else: print('##␣Need␣to␣align␣frequencies:␣TP:␣{0};␣Interf.:␣{1}'.format(
   ↪ tp_restfreq,int_restfreq))
imreframe(imagename=TPim,output=TPim+'.ref',outframe='lsrk',restfreq='
   ↪ 115271199999.99998Hz')
tp_restfreq=imhead(TPim+'.ref',mode='get',hdkey='restfreq') #check again
if tp_restfreq == int_restfreq: print('##␣Frequencies␣match')
else: print('##␣Need␣to␣align␣frequencies:␣TP:␣{0};␣Interf.:␣{1}'.format(
   ↪ tp_restfreq,int_restfreq))

#Regrid TP image to match Interferometry image (note that the 'Freq' and 'Stokes'
   ↪  axes are flipped)
imregrid(imagename=TPim+'.ref',
        template=Intim,
        axes=[0,1,2],
        output='TP.image.regrid')


#Trim the 7m+12m and (regridded) TP images
```

```
#Do this with the mask that was created in TCLEAN based on the PB (can also do
   ↪ with a box, as in CASAGuides)
os.system('rm␣-rf␣TP.regrid.subim')
imsubimage(imagename='TP.image.regrid',
          outfile='TP.regrid.subim',
          mask=Intmask+'.cut',stretch=True)

os.system('rm␣-rf␣Int.image.subim')
imsubimage(imagename=Intim,
          outfile='Int.image.subim',
          mask=Intmask+'.cut',stretch=True)

#CONTINUE WITH TP.regrid.subim and Int.image.subim

#Mask the PB image to match the Int/TP images
imsubimage(imagename=Intpb+'.cut',
          outfile=Intpb+'.subim',
          mask=Intmask+'.cut')

#Multiply the TP image by the 7m+12m primary beam response
os.system('rm␣-rf␣TP.subim.depb')
immath(imagename=['TP.regrid.subim',
                  Intpb+'.subim'],
       expr='IM0*IM1',stretch=True,
       outfile='TP.subim.depb')

#I see this warning, but ignore it: 2019-08-14 07:42:02 WARN ImageExprCalculator
   ↪ ::compute image units are not the same: 'Jy/beam' vs ''. Proceed with
   ↪ caution. Output image metadata will be copied from one of the input images
   ↪ since imagemd was not specified



#########################
## Ready to FEATHER
#########################

os.system('rm␣-rf␣Feather.image')
feather(imagename='Feather.image',
        highres='Int.image.subim',
        lowres='TP.subim.depb')

## Follow up with some test of whether flux is recovered; compare with SD-only
   ↪ image.
## Compare Feather.image and TP.subim.depb and 'TPForComb.dropdeg'

imstat('Feather.image')['flux']
imstat('TP.subim.depb')['flux']
```

```
## Not sure what units that is in. Beams should be difference, but cell size is
   ↪ the same (I think).

## Make an image of "fidelity" (model/(model-image))
os.system('rm -rf Feather.fidelity')
immath(imagename=['Feather.image',
                  'TP.subim.depb'],
       expr='IM1/(IM1-IM0)',stretch=True,
       outfile='Feather.fidelity')

#######################
## Day 2: TP2VIS
## Documentation here: https://github.com/tp2vis/distribute
#########################

'''
#1.0 Collect the files
#You might need this, in case you started again.
datadir = 'path/DataComb2019/Lup3mms/Lup3mms_Share/' #the directory where your
   ↪ test data are held
vis7m = datadir+'mst_07_nchan10_start0kms.ms'
vis12m = datadir+'mst_12_nchan10_start0kms.ms'
TPfits = datadir+'TP_12CO.fits'
TPim = 'TP.image' #You choose the name of your TP image
#importfits(fitsimage=TPfits,imagename=TPim) ## You just need to do this the
   ↪ first time, in order to get your TP Fits file into the CASA format (*.image
   ↪ )
vis12m7m = 'int12m7m.ms' #You choose the name of your combined interferometry
   ↪ image
'''

## IN PROGRESS

#######################
## Day 3: SDInt
## Documentation here: https://github.com/urvashirau/WidebandSDINT
#########################

## IN PROGRESS
```