Pipelines for radio interferometric data reduction

Ruta Kale
National Centre for Radio Astrophysics,
Tata Institute of Fundamental Research,
Pune, India

Radio Astronomy School 2019, 30th August 2019

Outline

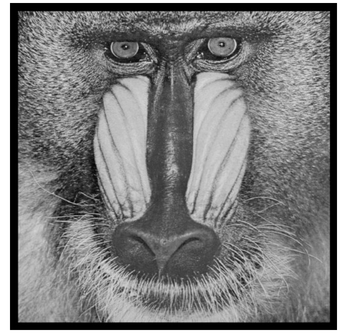
- Why pipelines ?
- AIPS based pipelines
- Writing your own AIPS pipeline
- CAPTURE : CASA Pipeline-cum-Toolkit for UGMRT Data Reduction
- Writing tasks in CASA, creating your own pipelines

Why pipelines ?

- Data reduction is a lengthy process, many parts are similar across datasets: automation !
- Increases reproducibility of the results
- Reduces human errors
- Data sizes are large or going to get larger: easy to port to servers and run remotely.
- Ease of testing one aspect at a time for complex algorithms like “tclean”
- A step towards “open science”: https://zenodo.org/record/2631868#.XWiX-JzhVUQ

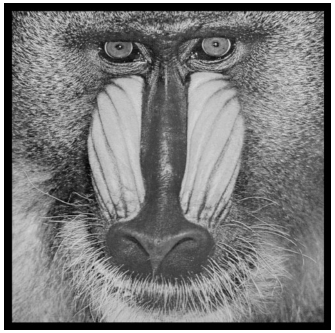
  “Reproducibility and open science in the SKA era” by Rachel Ainsworth

AIPS



- RUNFILES
- Commands can be put into a text file and the text file can be provided to AIPS
- Quirks: needs to have extension of AIPS userid in e-hex format
- Any programming: e. g. for loops, condition testing etc. in “Parseltongue”

AIPS



- SPAM: Source Peeling and Atmospheric Modeling (Intema et al 2009)
- http://www.intema.nl/doku.php?id=huibintemaspam

CASA

- Writing tasks and pipelines: Python
- Writing a task in CASA:

  XML file : sets the input interface

  taskname.py : actual code
- Use of CASA tasks and toolkit functionalities
- Excellent documentation for learning.

CASA

- Writing tasks and pipelines: Python
- Pipeline:

  - automating the process that you do interactively at the terminal

  - automation of decision making is crucial !

CAPTURE

CASA Pipeline-cum-Toolkit for UGMRT Data Reduction

- Automation of routine processes: conversion from lta to MS, flagging of bad antennas, standard calibration, splitting target source data.
- Efficient elimination of RFI while not overdoing it.
- Automated self-calibration but still giving enough freedom to the to choose the strategy.
- Easily tailored for special needs: for e. g. for online RFI excision system testing: can deal with only calibrator data, half or one fourth of an array of data

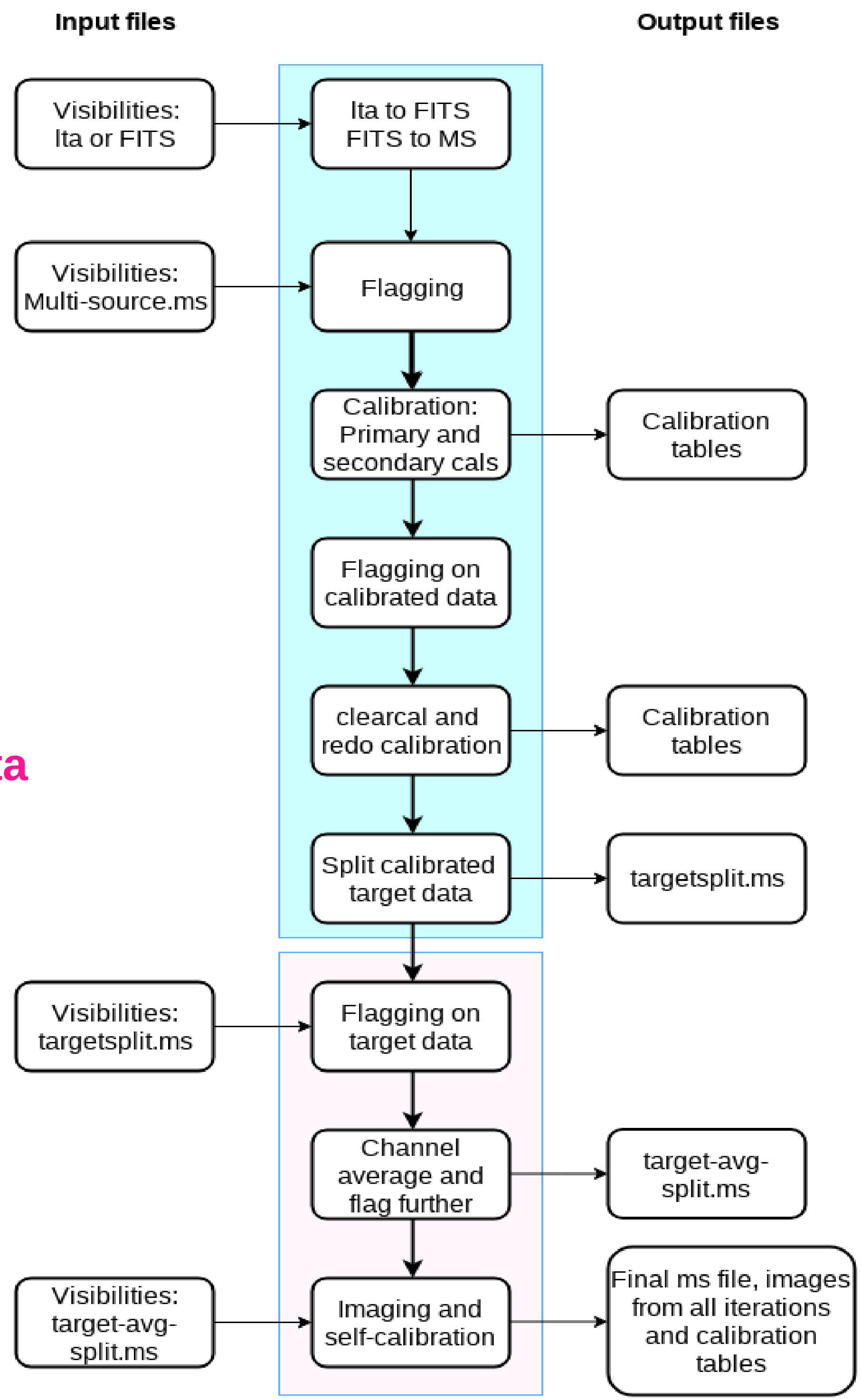
CAPTURE

https://github.com/ruta-k/uGMRT-pipeline

Python 2.7 and Common Astronomy Software Applications

(CASA, McMullin, J. P. et al 2007)

CAPTURE



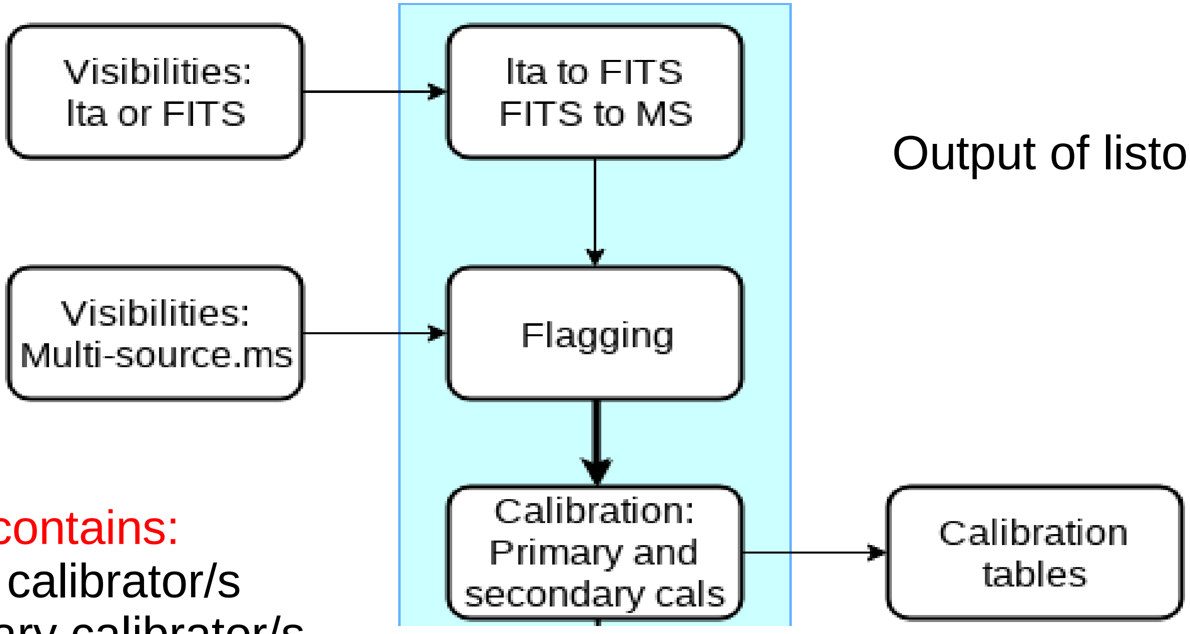
Multi-source data

- Initial flagging and calibration
- Further flagging and final calibration

Working with calibrated target source data

- flagging
- averaging in frequency
- imaging and self-calibration

Initial flagging and calibration

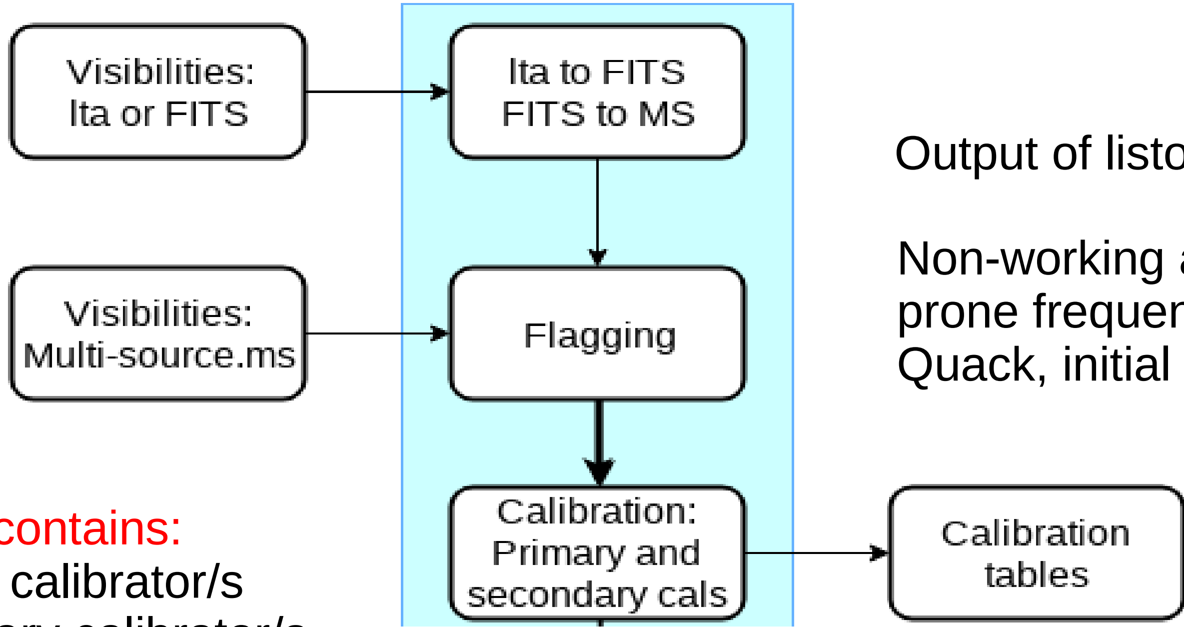


Output of listobs in a .list file

MS file contains:
Primary calibrator/s
Secondary calibrator/s
Target/s: source names not in the vla list of calibrators

Dependencies:
If starting from lta file: listscan, gvfits
Calibration part: vla-cals.list

Initial flagging and calibration



Output of listobs in a .list file

Non-working antennas, RFI prone frequency channels, Quack, initial clip and tfcrop

MS file contains:
Primary calibrator/s
Secondary calibrator/s
Target/s: source names not in the vla list of calibrators

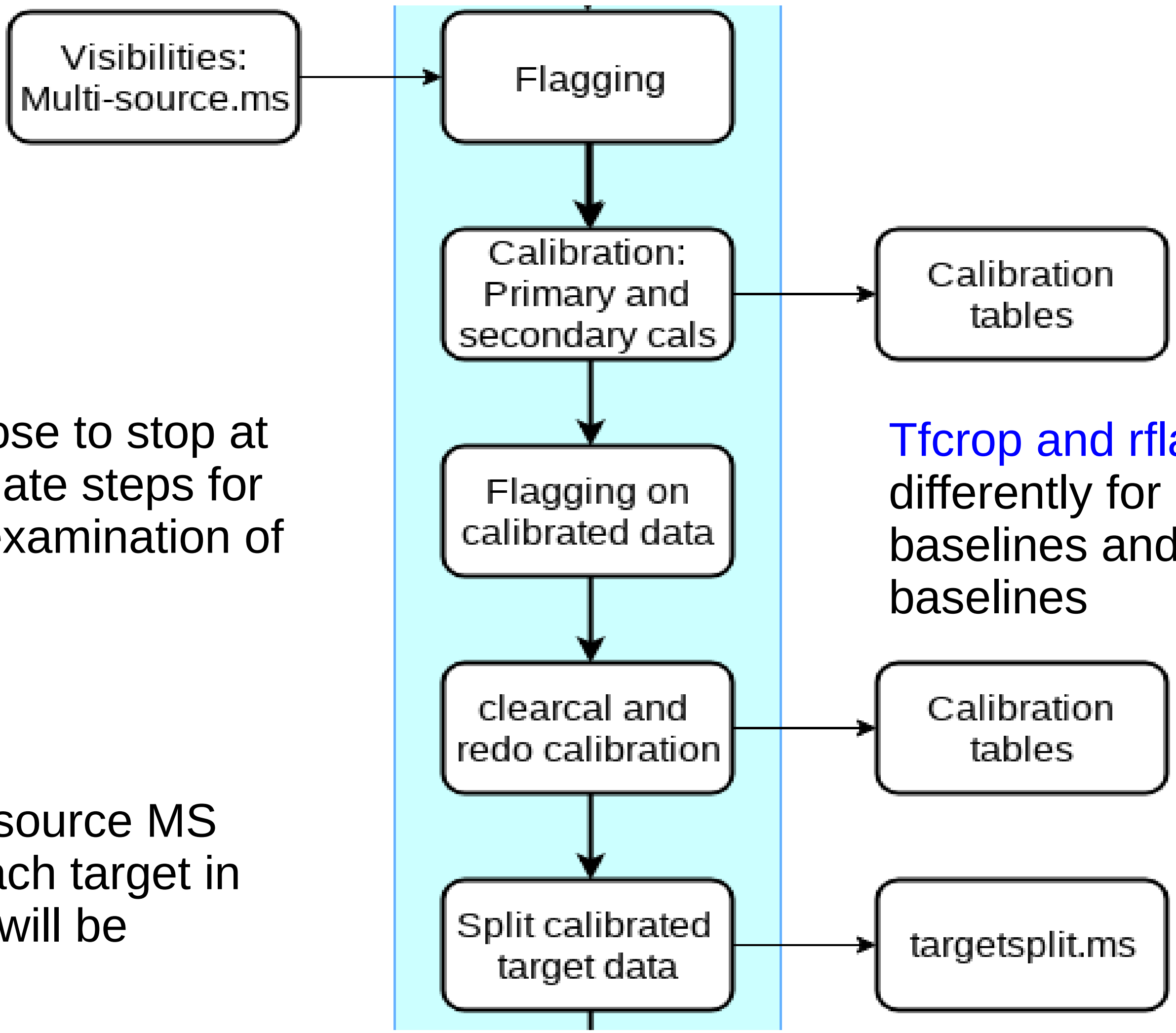
Gain tables: K, G
Bandpass table
Fluxscale table

Dependencies:
If starting from lta file: listscan, gvfits
Calibration part: vla-cals.list

Further flagging and calibration

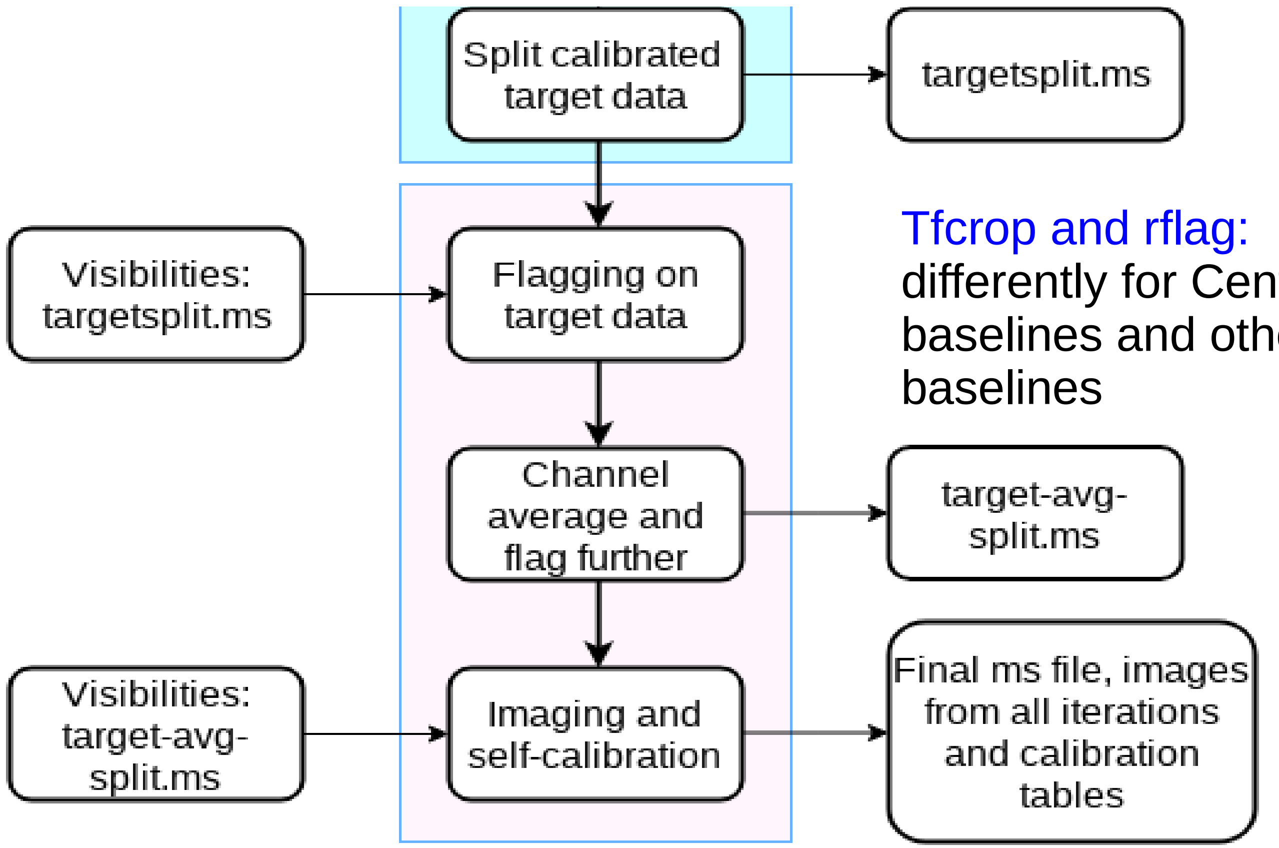
Can choose to stop at intermediate steps for manual examination of data.

A single source MS file for each target in the data will be created.



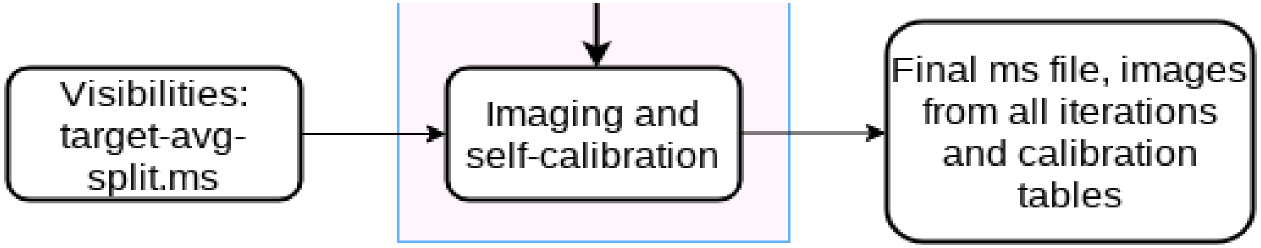
Tfcrop and rflag: differently for Central sq. baselines and other baselines

Working with calibrated target data



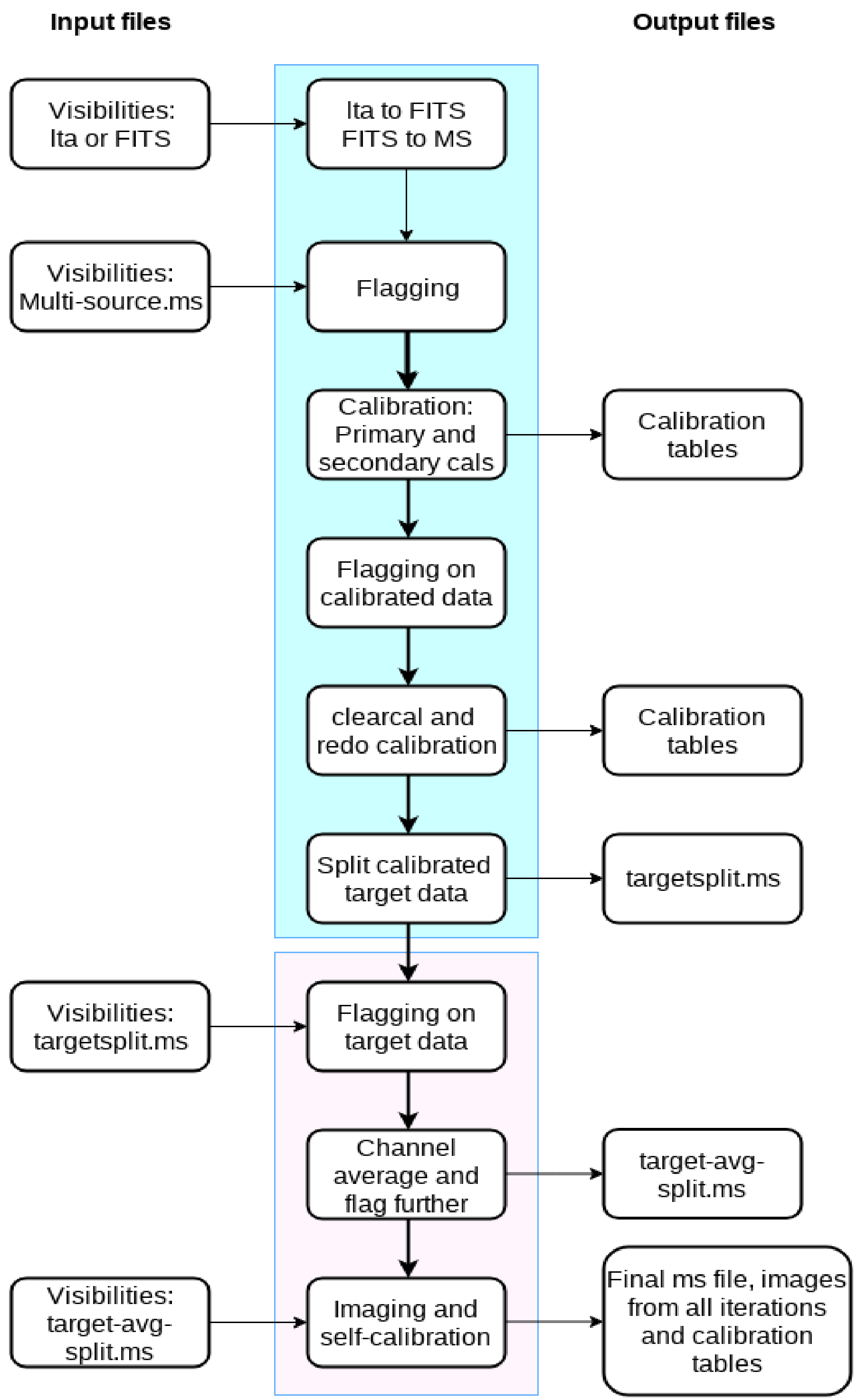
Tfcrop and rflag:
differently for Central sq. baselines and other baselines

Working with calibrated target data



Imaging and self-calibration:

- Option to make a dirty image – can examine and decide self-calibration strategy.
- Phase-only and amp and phase self-calibration iterations given by the user carried out.
- Flagging on residual data column is carried out in the self-calibration loop.


Input files
Output files
Visibilities: lta or FITS
lta to FITS FITS to MS
Visibilities: Multi-source.ms
Flagging
Calibration: Primary and secondary cals
Calibration tables
Flagging on calibrated data
clearcal and redo calibration
Calibration tables
Split calibrated target data
targetsplit.ms
Visibilities: targetsplit.ms
Flagging on target data
Channel average and flag further
target-avg-split.ms
Visibilities: target-avg-split.ms
Imaging and self-calibration
Final ms file, images from all iterations and calibration tables

Structure of the pipeline program

The pipeline is a single python program.

Initial set-up

Inputs

These need to be modified according to the type of the data that are to be analysed.

Functions

Applications

Modifications only for any special requirements.

Initial set-up: initial flagging and calibration

```
###### SET THE STAGE FOR DATA ANALYSIS ##############################
fromlta = True                        # If starting from lta file set it True.
gvbinpath = ['./listscan','./gvfits'] # set the path to listscan and gvfits if fromlta==True.
fromraw = True                        # True if starting from FITS data. Otherwise keep it False.
fromms = True                         # True If working with multi-source MS file.
findbadants = True                    # find bad antennas when True
flagbadants= True                     # find and flag bad antennas when True
findbadchans = True                   # find bad channels within known RFI affected freq ranges when True
flagbadfreq= True                     # find and flag bad channels within known RFI affected freq ranges when True
myflaginit = True                     # True to flag first channel, quack, initial clips
doinitcal = True                      # True to calibrate data
mydoflag = True                       # True to flag on the calibrated data
redocal = True                        # True to redo calibration - recommended
dosplit = True                        # True to split calibrated data on target source
mysplitflag = True                    # True to flag on the target source
dosplitavg = True                     # True to average channels
doflagavg = True                      # True to flag on the channel averaged file
makedirty = True                      # True only if you want to make a dirty image of your target source
doselfcal = True                      # True if selfcal loop should be run
usetclean = True                      # True if you want to use tclean (recommended); False will use clean.
```

Initial set-up: input files

```
###### SET THE STAGE FOR DATA ANALYSIS #############################

fromlta = True                                # If starting from lta file set it True.

gvbinpath = ['./listscan','./gvfits'] # set the paths to listscan, gvfits

fromraw = True                                # True if starting from FITS data.

fromms = True                                 #True if working with multi-source MS
```

Initial set-up: initial flagging and calibration

```
###### SET THE STAGE FOR DATA ANALYSIS ##############################
fromlta = True                          # If starting from lta file set it True.
gvbinpath = ['./listscan','./gvfits'] # set the path to listscan and gvfits if fromlta==True.
fromraw = True                          # True if starting from FITS data. Otherwise keep it False.
fromms = True                           # True If working with multi-source MS file.
findbadants = True                      # find bad antennas when True
flagbadants= True                       # find and flag bad antennas when True
findbadchans = True                     # find bad channels within known RFI affected freq ranges when True
flagbadfreq= True                       # find and flag bad channels within known RFI affected freq ranges when True
myflaginit = True                       # True to flag first channel, quack, initial clips
doinitcal = True                        # True to calibrate data
mydoflag = True                         # True to flag on the calibrated data
redocal = True                          # True to redo calibration - recommended
dosplit = True                          # True to split calibrated data on target source
mysplitflag = True                      # True to flag on the target source
dosplitavg = True                       # True to average channels
doflagavg = True                        # True to flag on the channel averaged file
makedirty = True                        # True only if you want to make a dirty image of your target source
doselfcal = True                        # True if selfcal loop should be run
usetclean = True                        # True if you want to use tclean (recommended); False will use clean.
```

Initial set-up: initial flagging and calibration

SET THE STAGE FOR DATA ANALYSIS

findbadants = True # find bad antennas when True

flagbadants= True # find and flag bad antennas when True

findbadchans = True # find bad channels within known RFI affected freq ranges when True

flagbadfreq= True # find and flag bad channels within known RFI affected freq ranges when True

myflaginit = True # True to flag first channel, quack, initial clips

doinitcal = True # True to calibrate data

Initial set-up: flagging, calibration and split

```
###### SET THE STAGE FOR DATA ANALYSIS ###############################
fromlta = True                          # If starting from lta file set it True.
gvbinpath = ['./listscan','./gvfits'] # set the path to listscan and gvfits if fromlta==True.
fromraw = True                          # True if starting from FITS data. Otherwise keep it False.
frommms = True                          # True If working with multi-source MS file.
findbadants = True                      # find bad antennas when True
flagbadants= True                       # find and flag bad antennas when True
findbadchans = True                     # find bad channels within known RFI affected freq ranges when True
flagbadfreq= True                       # find and flag bad channels within known RFI affected freq ranges when True
myflaginit = True                       # True to flag first channel, quack, initial clips
doinitcal = True                        # True to calibrate data
mydoflag = True                         # True to flag on the calibrated data
redocal = True                          # True to redo calibration - recommended
dosplit = True                          # True to split calibrated data on target source
mysplitflag = True                      # True to flag on the target source
dosplitavg = True                       # True to average channels
doflagavg = True                        # True to flag on the channel averaged file
makedirty = True                        # True only if you want to make a dirty image of your target source
doselfcal = True                        # True if selfcal loop should be run
usetclean = True                        # True if you want to use tclean (recommended); False will use clean.
```

Initial set-up: flagging, calibration and split

SET THE STAGE FOR DATA ANALYSIS

```
mydoflag = True          # True: flags on the calibrated data

redocal = True           # True to redo calibration

dosplit = True           # True to split calibrated target data
```

Initial set-up: frequency avg, flagging

```
###### SET THE STAGE FOR DATA ANALYSIS ##############################
fromlta = True                          # If starting from lta file set it True.
gvbinpath = ['./listscan','./gvfits'] # set the path to listscan and gvfits if fromlta==True.
fromraw = True                          # True if starting from FITS data. Otherwise keep it False.
fromms = True                           # True If working with multi-source MS file.
findbadants = True                      # find bad antennas when True
flagbadants= True                       # find and flag bad antennas when True
findbadchans = True                     # find bad channels within known RFI affected freq ranges when True
flagbadfreq= True                       # find and flag bad channels within known RFI affected freq ranges when True
myflaginit = True                       # True to flag first channel, quack, initial clips
doinitcal = True                        # True to calibrate data
mydoflag = True                         # True to flag on the calibrated data
redocal = True                          # True to redo calibration - recommended
dosplit = True                          # True to split calibrated data on target source
mysplitflag = True                      # True to flag on the target source
dosplitavg = True                       # True to average channels
doflagavg = True                        # True to flag on the channel averaged file
makedirty = True                        # True only if you want to make a dirty image of your target source
doselfcal = True                        # True if selfcal loop should be run
usetclean = True                        # True if you want to use tclean (recommended); False will use clean.
```

Initial set-up: frequency avg, flagging

SET THE STAGE FOR DATA ANALYSIS

```
mysplitflag = True          # True to flag on the target source

dosplitavg = True           # True to average channels

doflagavg = True            # True to flag on the channel
averaged file
```

Initial set-up: imaging and self-calibration

SET THE STAGE FOR DATA ANALYSIS

makedirty = True # True only if you want to make a dirty image of your target source

doselfcal = True # True if selfcal loop should be run

usetclean = True # True if you want to use tclean (recommended); False will use clean.

Inputs

```
###### INPUTS ###########################################################
ltafile =''                        # lta file
rawfile = ''                      # TEST.FITS or provide the name of the FITS file if you already have;
myfile1 =''                        # MS file   (REQUIRED if starting from multi-source MS file)
mysplitfile =''                 # target source file name (split file)
mysplitavgfile = ''          # target source file name after averaging; REQUIRED if starting from this file
# Inputs for flagging and calibration
myquackinterval = 10.0     # time in s to flag at the beginning of a scan and at the end of the scan.
clipfluxcal =[0.0,60.0]       # in Jy. typically twice the expected flux; only to remove high points
clipphasecal =[0.0,60.0]   # in Jy. typically twice the expected flux; only to remove high points
cliptarget =[0.0,30.0]         # in Jy. typically four times the expected flux; only to remove high points
clipresid=[0.0,10.0]           # in Jy. 10 times the rms for single channel and single baseline
myrefant = 'C00'                # choose a reference antenna - make sure it is one of the working antennas.
uvracal =''                          # Leave it to ''; will apply it to all the calibrators in the current version of the pipeline
# Inputs for post split averaging of channels
mywidth2 = 10                   # number of channels to average - choose aptly to avoid bandwidth smearing.
# Inputs for imaging and self-calibration : You will need to change relevant advanced controls if you change the values here.
scaloops = 8                     # Total number of self-cal loops (including both phase-only and amp-ph)
mypcaloops = 4                # Number of p-only selfcal loops; should be <= scaloops. The remaning loops will and a&p self-cal.
mythresholds = 0.1          # in mJy. Global flux threshold – starting threshold – will change with iteration.
mycell = ['2.0arcsec']      # Set the cellsize for imaging.
myimsize = [12000]         # Set the size of the image in pixel units. Should cover the primary beam.
```

More Inputs

Further control on imaging and self-calibration

mynterms = 2 # nterms used in tclean; not tested for nterms >2.

mywproj2 = -1 # Number of wprojection planes- leave it to -1 so that it is determined internally in tclean

Solint used for self-cal: provide solints for each self-cal iteration : edit according to the number of self-cal loops you

have chosen. Has to be of the same length as nscaloops

mysolint2 = ['8.0min','4.0min','2.0min','1.0min','8.0min','4.0min','2.0min','1.0min']

uvrascal='' # uvrange cutoff used in self-calibration – will use in the task gaincal.

Structure of the pipeline program

The pipeline is a single python program.

Initial set-up Inputs	These need to be modified according to the type of the data that are to be analysed.
Functions	A list of python function calling CASA tool-kit and tasks
Applications	Main processing block: runs the functions with the inputs given by the user.

How to run the pipeline ?

- Works in CASA versions 5.0 and above. Likely also in earlier ones but not tested.
- Copy all the files from github to the directory from which you are going to run the pipeline.
- In the python program file (.py), make the “Initial set-up” - set the True or False states of the parameters and then provide the “Inputs”.

Save and run the file using:

casa -c capture-pipeline-V0.py

Or at the CASA prompt using:

execfile(“capture-pipeline-V0.py”)

Images obtained using the pipeline

Pipeline works for:

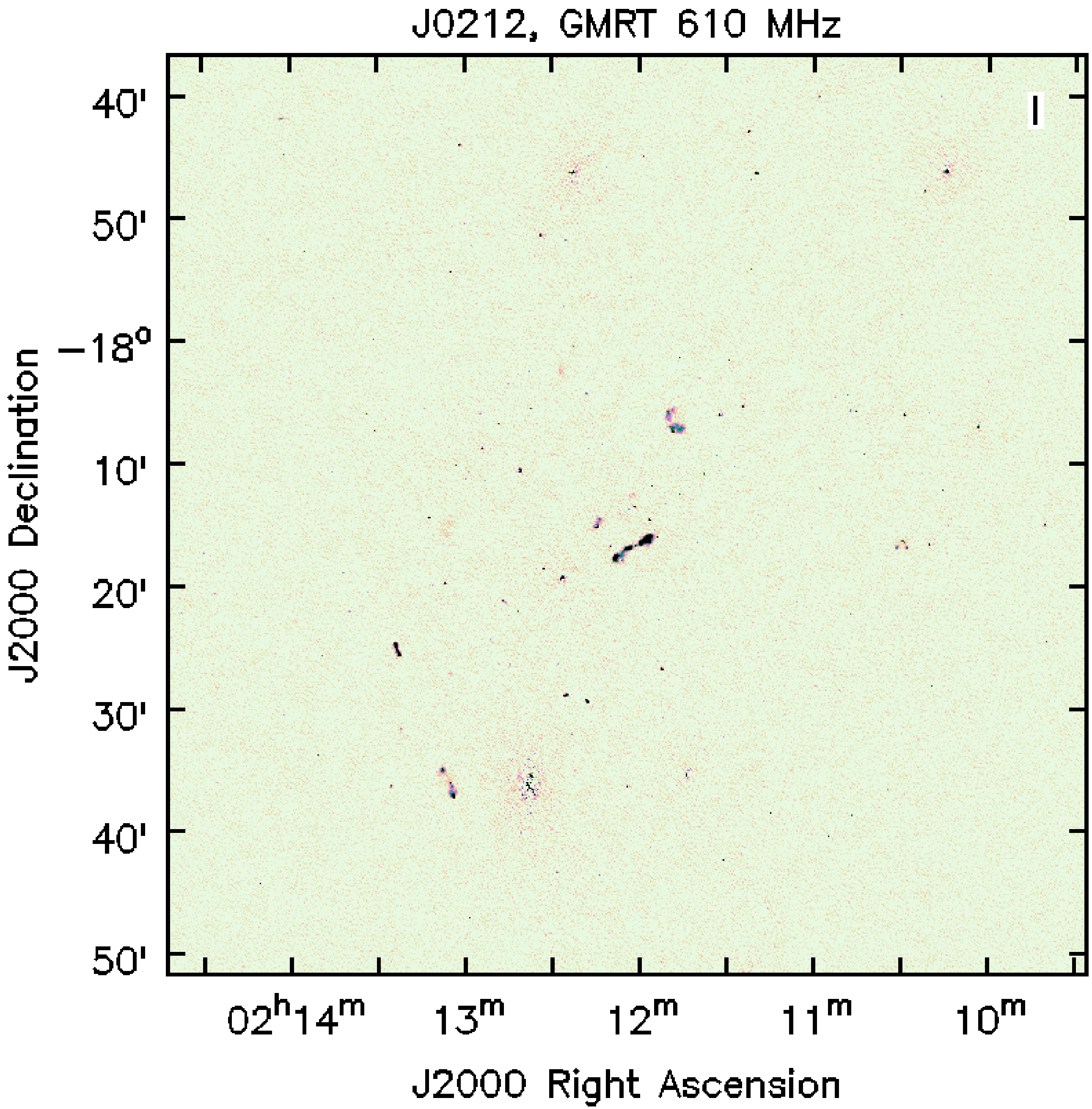
Legacy GMRT : All bands except data taken in dual frequency mode.

Upgraded GMRT: Bands 3, 4 and 5

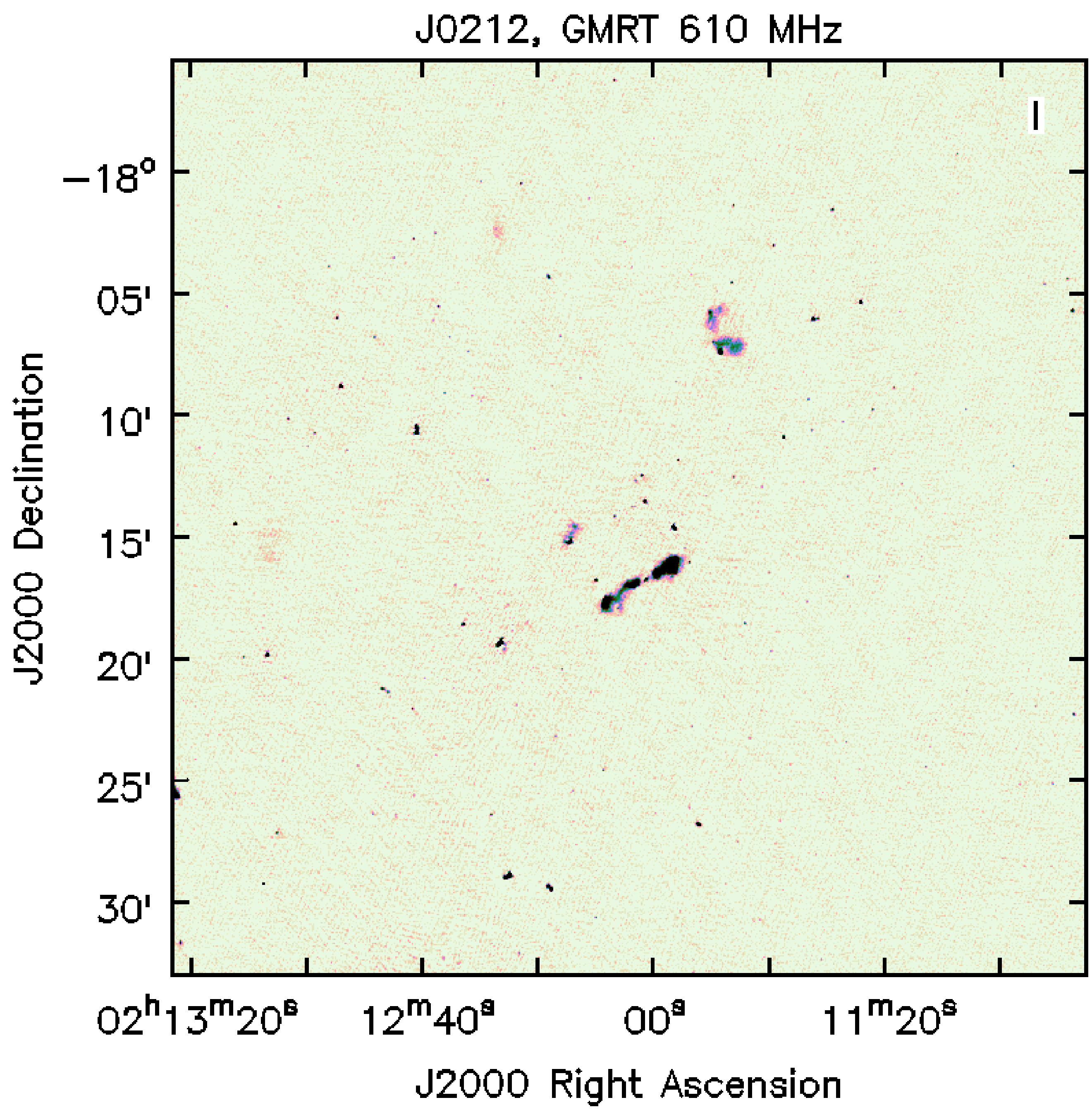
For Band 2: it works after editing the choice of channels to accommodate the notch filter.

Works on sub-array data as well.

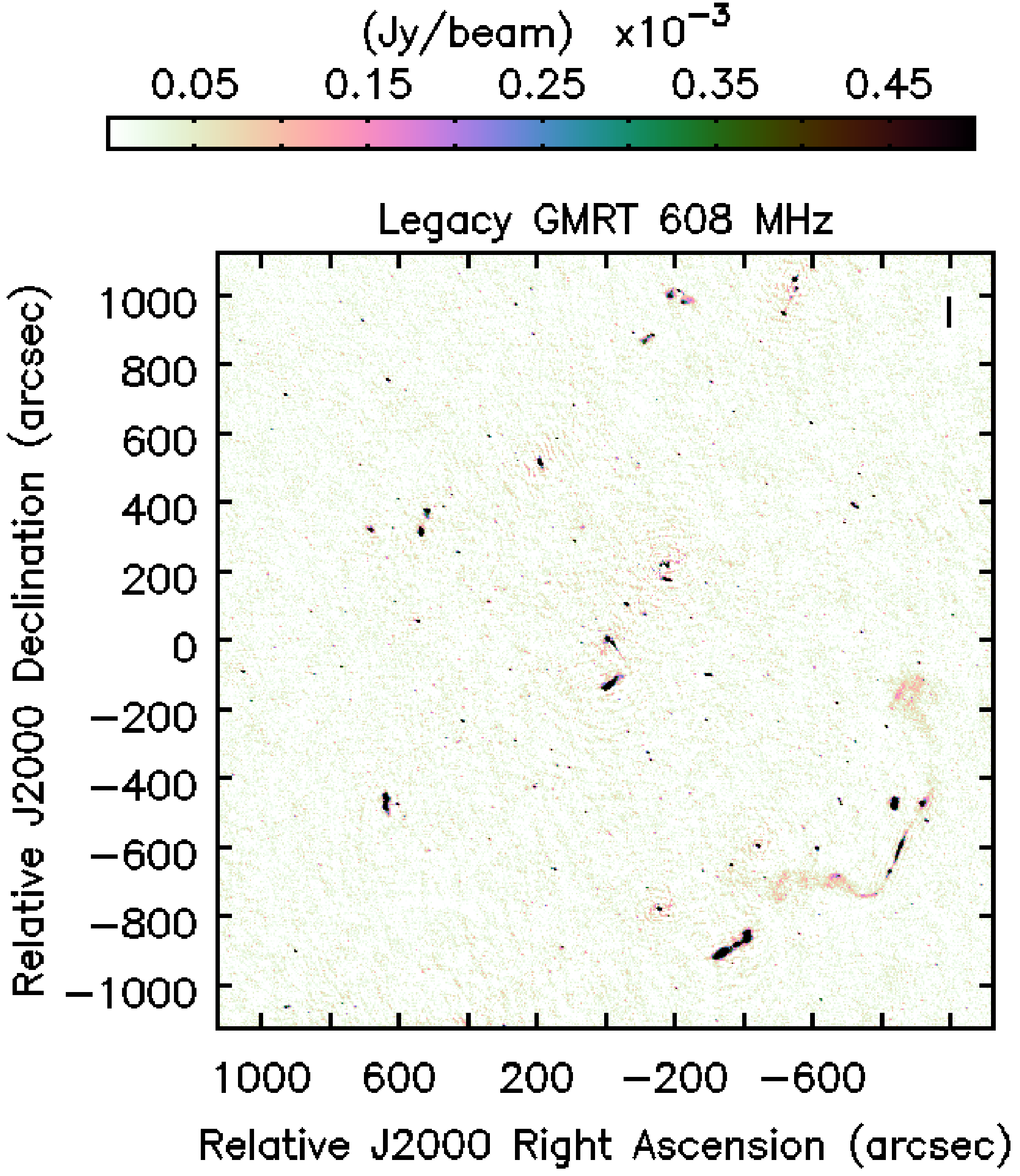
Works when flux and phase calibrator are the same.



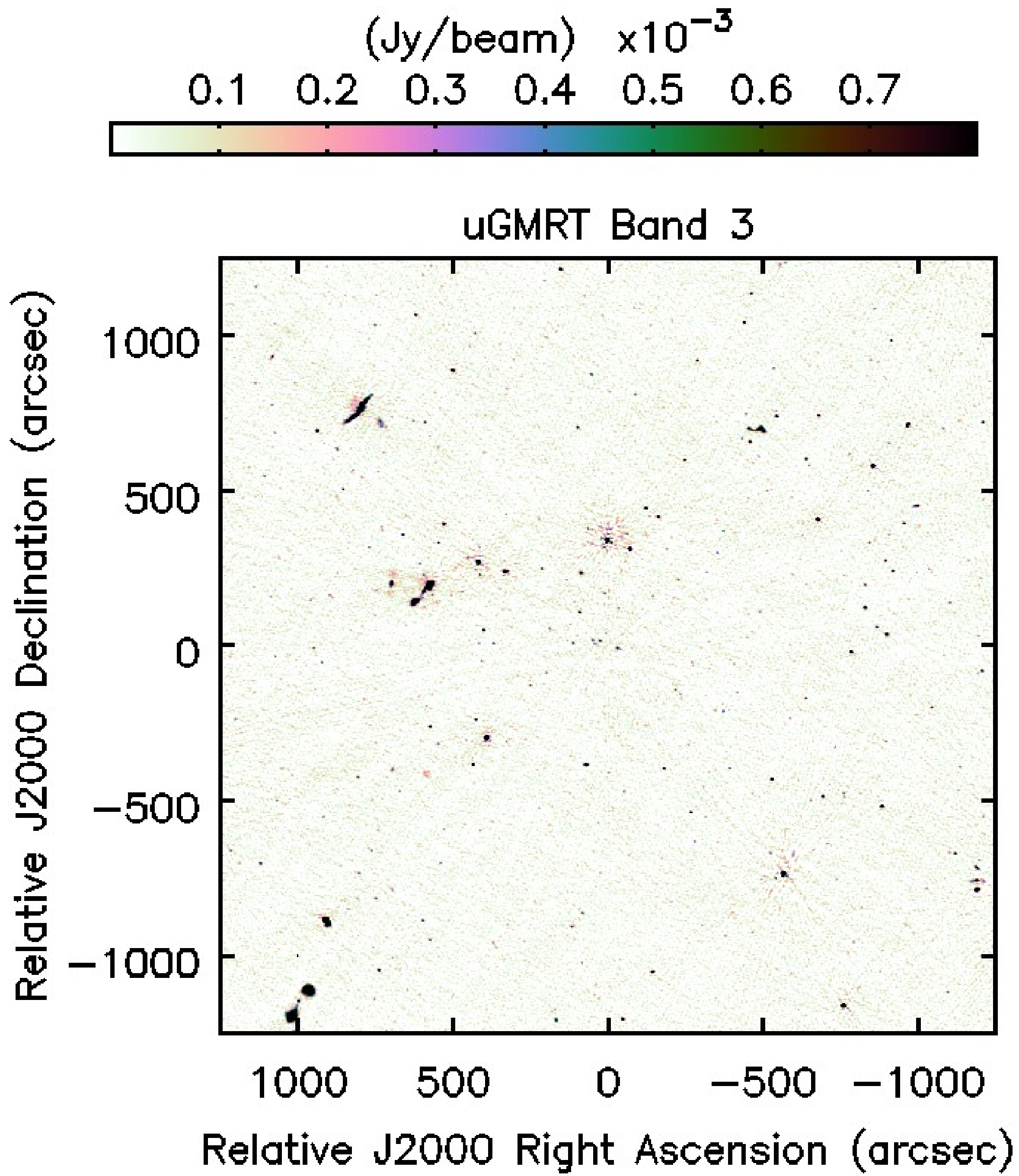
Peak 0.17 Jy/b
Rms 0.048 mJy/b



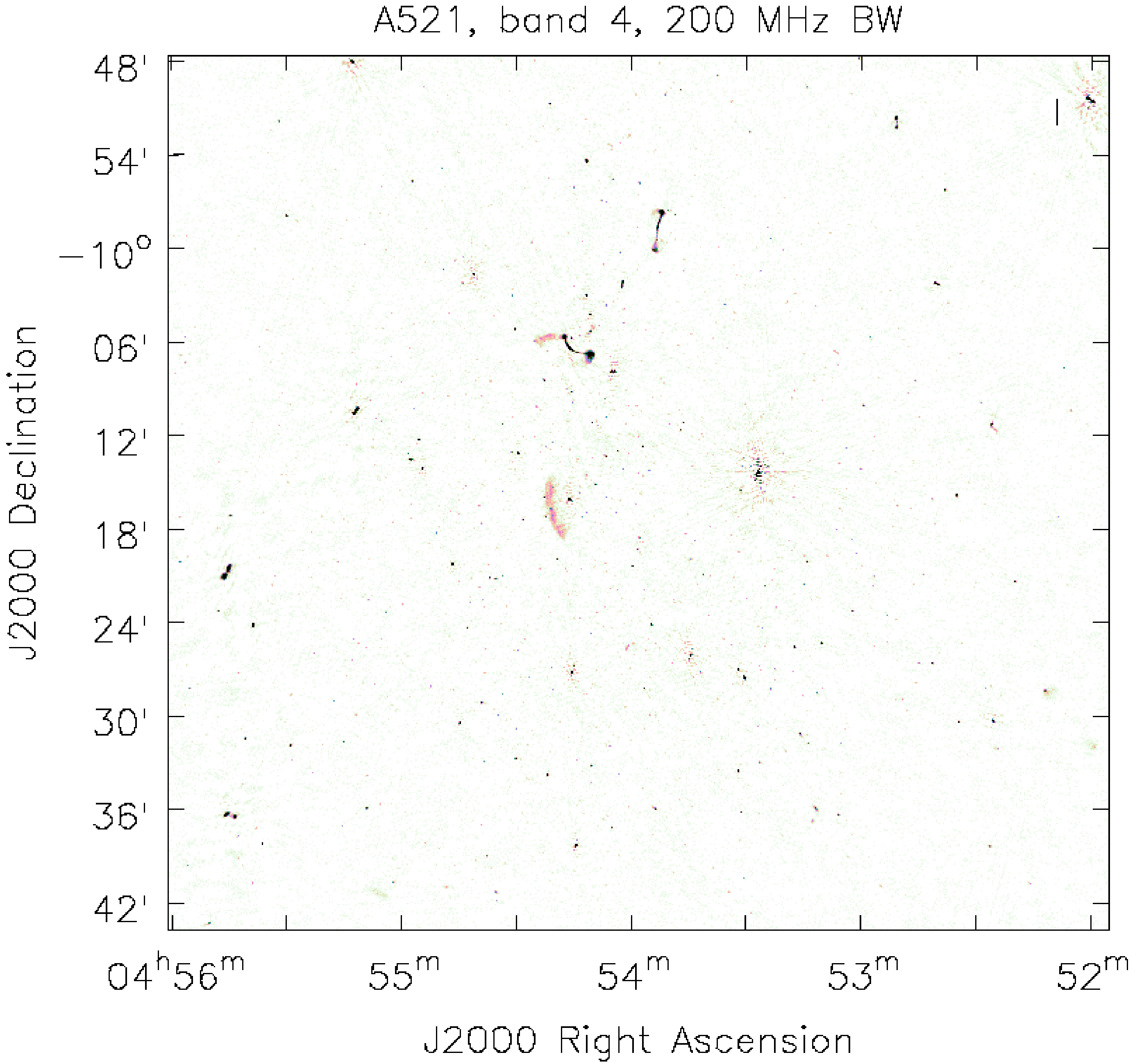
Peak 0.17 Jy/b
Rms 0.048 mJy/b



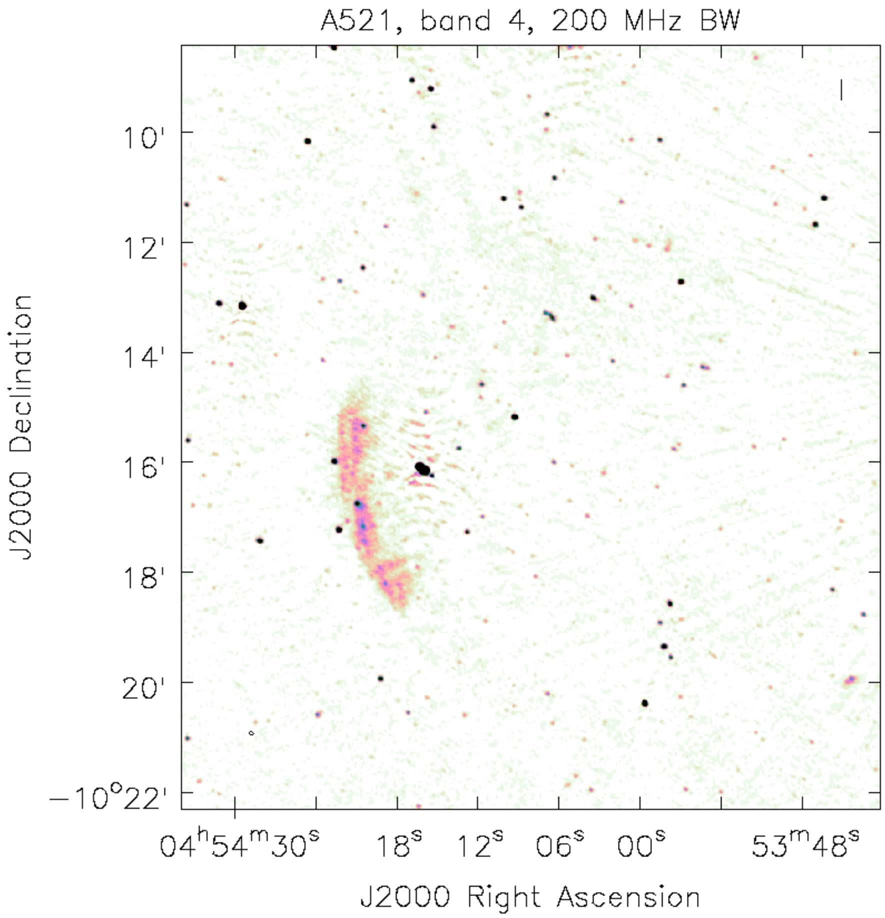
Peak 0.06 Jy/b
RMS 0.028 mJy/b

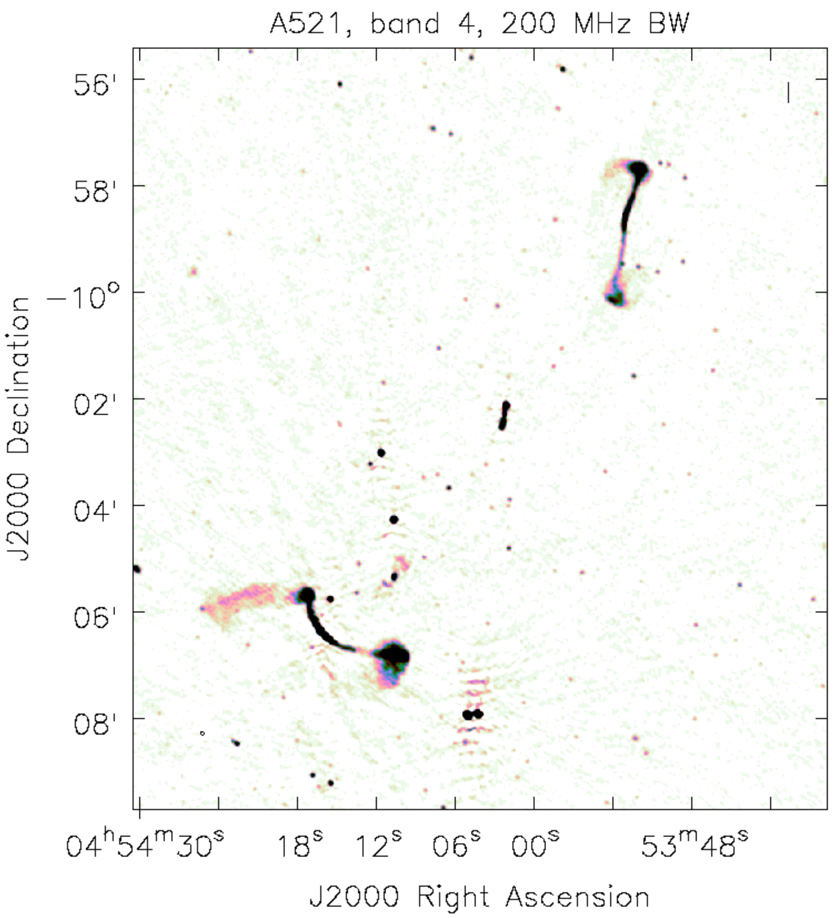


Peak 0.08 Jy/b
RMS 0.048 mJy/b



Band 4
Peak 0.045
RMS 0.01 mJy/b





Band 4
Peak 0.045 Jy/b
RMS 0.01 mJy/b

Pipeline run duration

Sample numbers:

uGMRT P-band data with on source time of 2.5 hours,
cellsize = 1 arcsec, Imsize = 10000 pixels.

Time taken from split file to final self-calibrated image:
32GB RAM, 8 processors, 3.4 GHz : 3.6 days

For a legacy dataset ~ 6 hr duration:
End-to-end ~3 days.

For uGMRT dataset ~ 9 hr duration, P-band:
End-to-end ~6 days on 128 GB, 24 cores (not an exclusive run)

Memory issues on 32GB machine if imagesize is large ~12000 or so.

CAPTURE: caveats

- Multiple targets: Two step run needed. First step to create calibrated split files for each source and then a separate imaging run for each target is needed.
- If a self-cal run is interrupted, the full imaging run needs to be carried out again.
- For a different choice of parameters in tclean, the python function “mytclean” can be edited appropriately and used (with caution!).
- The pipeline is not tested for nterms > 2.
- Full Stokes data reduction is possible but polarization calibration is not part of it as yet.
- Primary beam correction is a separate task to be run outside the pipeline. https://github.com/ruta-k/uGMRTprimarybeam

Summary

- Pipelines are essential: reproducibility, automation, data sizes, working on remote servers – in general makes life easier !
- Need to be used with caution as at low frequencies each field needs a tailored strategy to obtain the best possible image for the intended science.
- CAPTURE available for uGMRT continuum data reduction - can be easily tailored for special needs.
- Writing new CASA tasks, pipelines made easy by Python
- A move towards “open science”.