CHARIZARD, A New Radio Interferometric Pipeline for SKA Pathfinders and SKA

Arpan Pal

Collaborators: Ruta Kale

ICTS, 10th April 2025

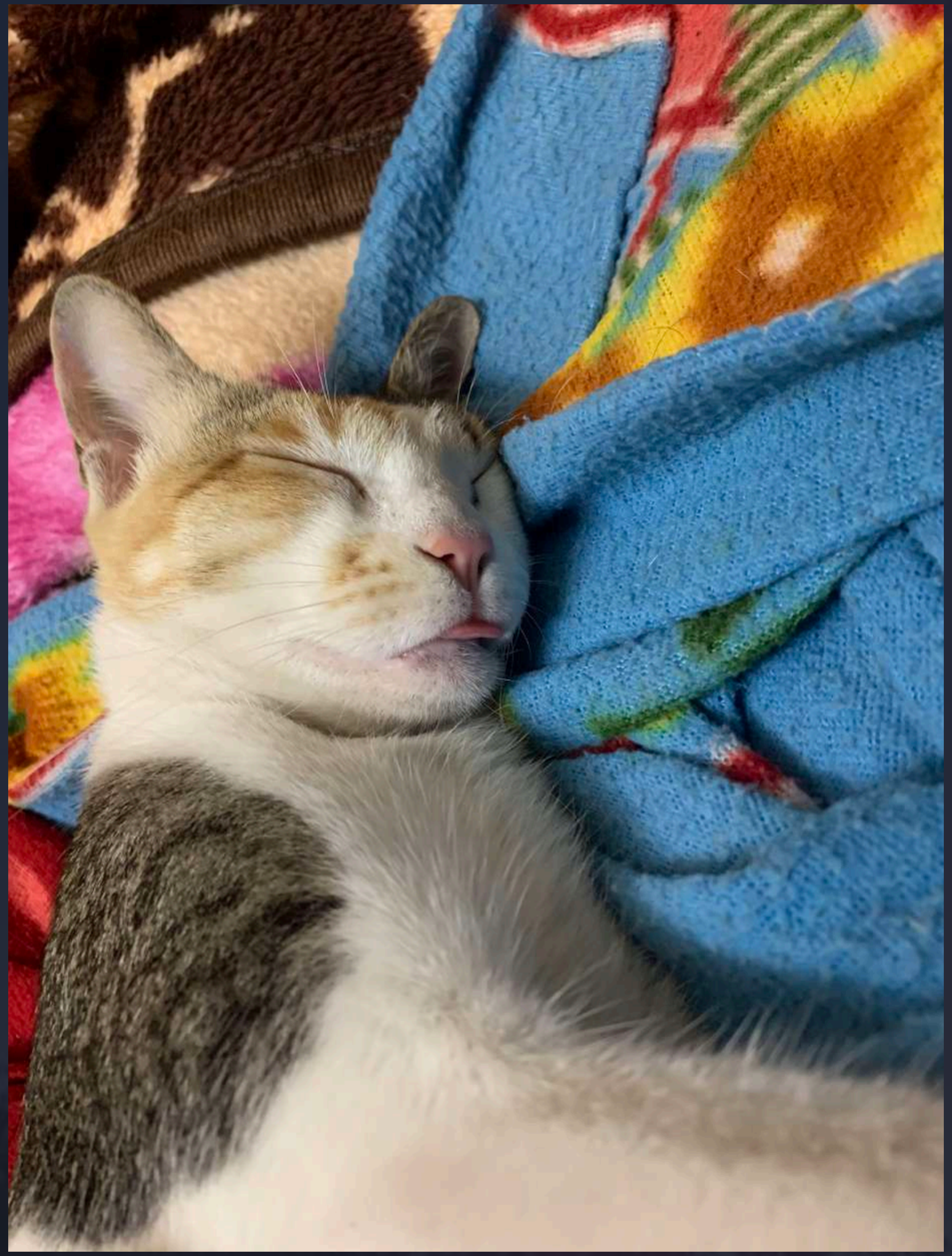
NCRA • TIFR

Pokemon image credits: pokeworld

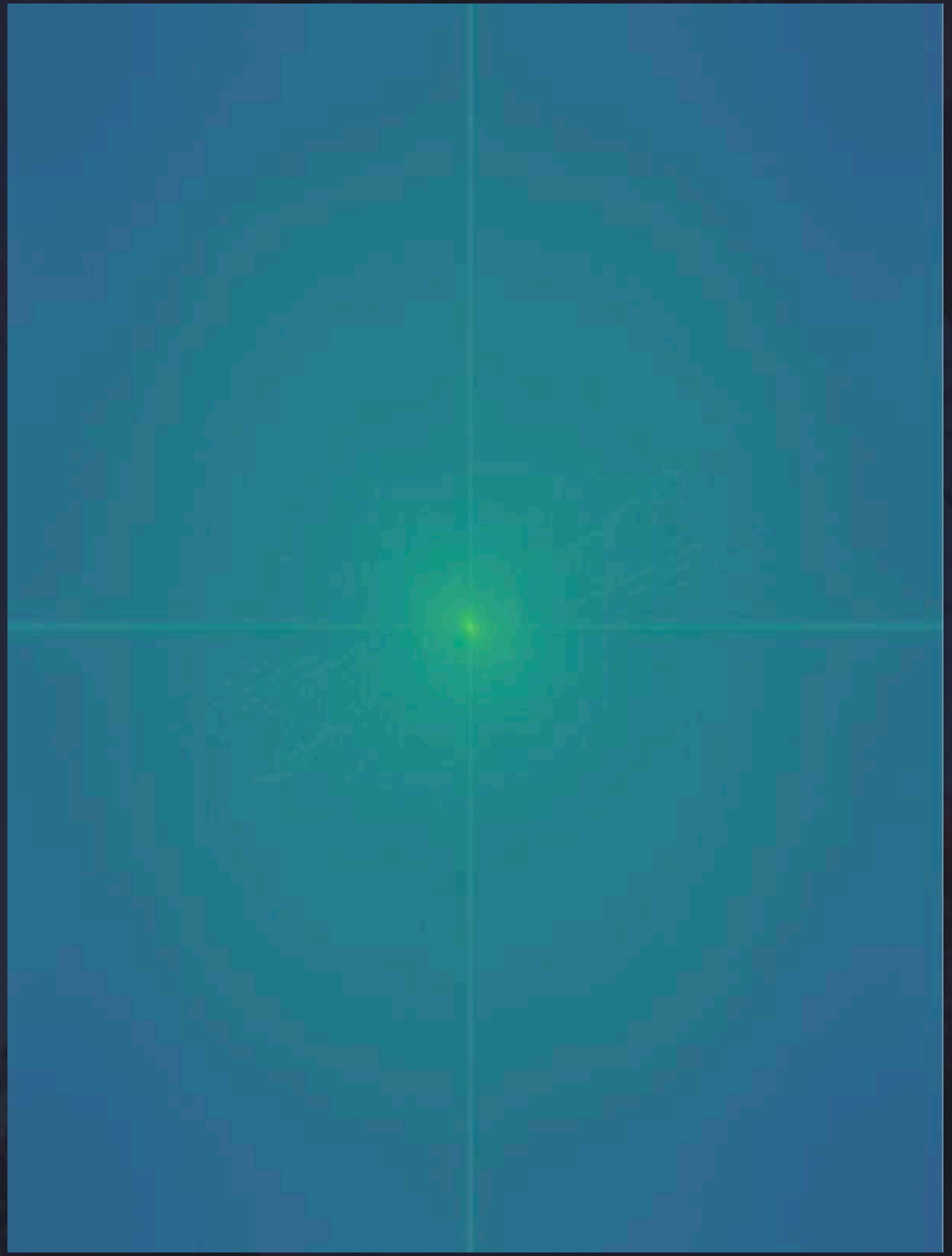
Many Cosmologists, so an Equation

$$V(u, v, w) = \int I(l, m) e^{-2i\pi(lu+mv+nw)} \frac{dldm}{\sqrt{1 - l^2 - m^2}}$$

Nimki



Nimki in Fourier Space



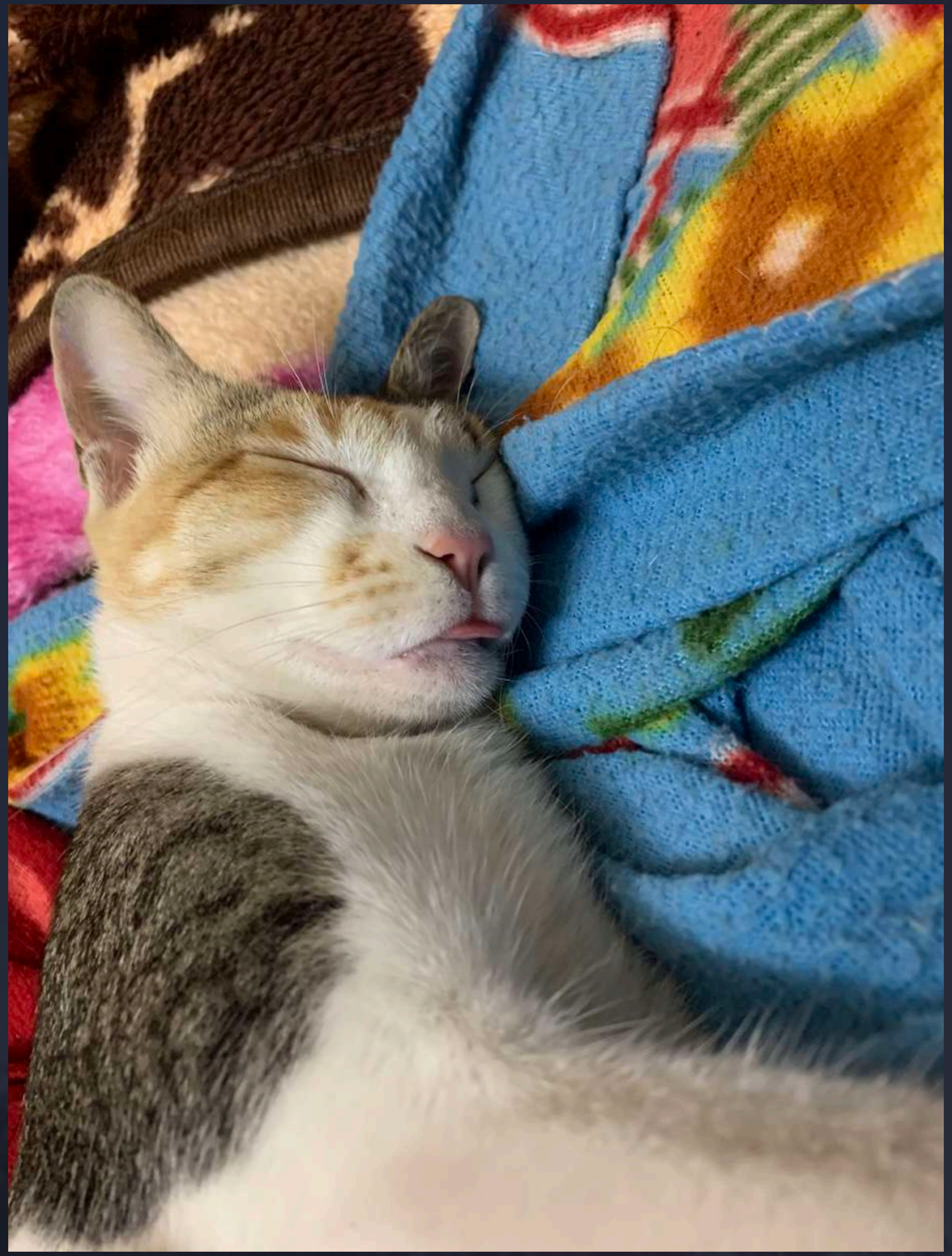
NCRA • TIFR

Many Cosmologists, so an Equation

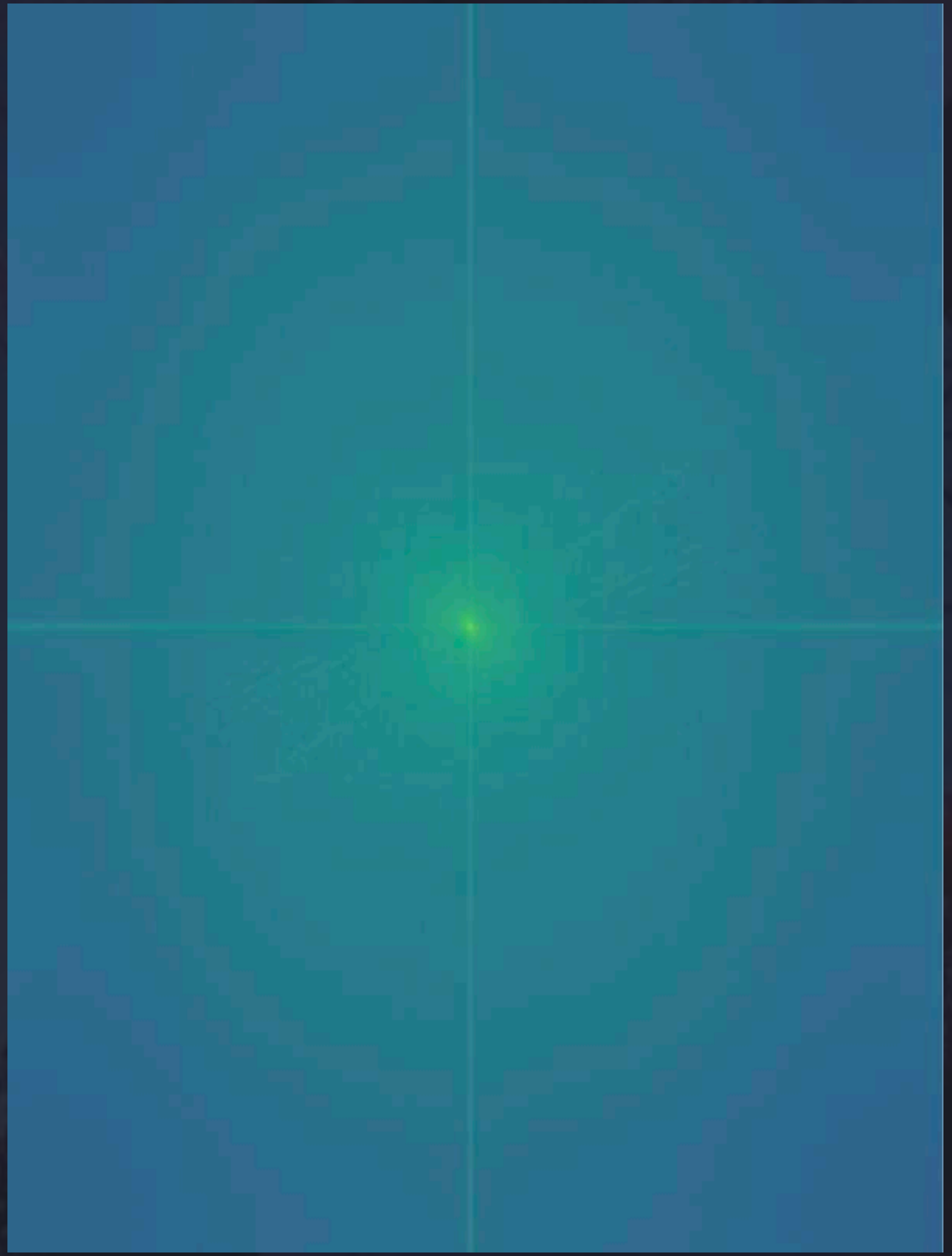
$$V(u,v,w) = \int I(l,m)e^{-2i\pi(lu+mv+nw)} \frac{dldm}{\sqrt{1-l^2-m^2}}$$

$$V_{observed} = MSV_{true}$$

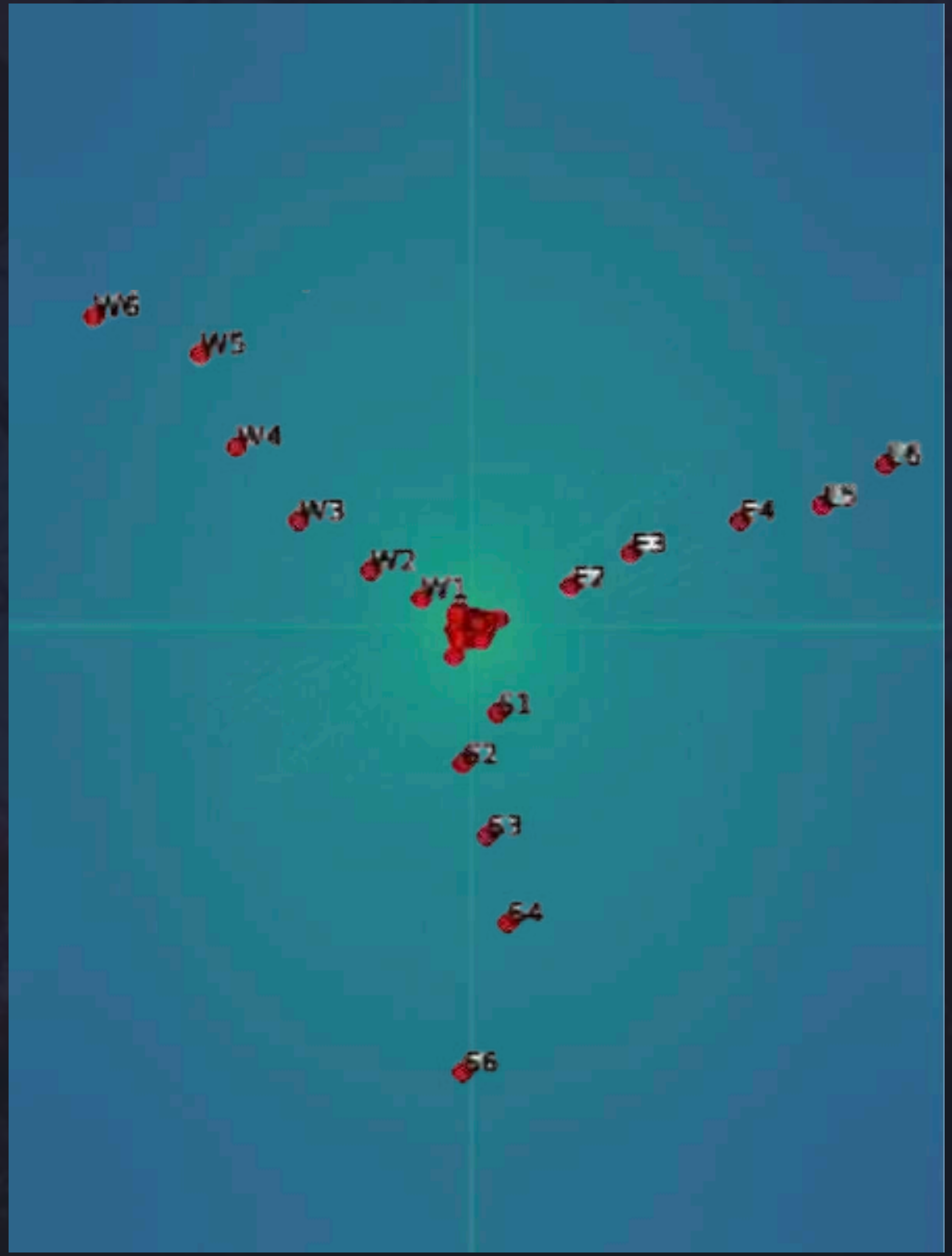
Nimki



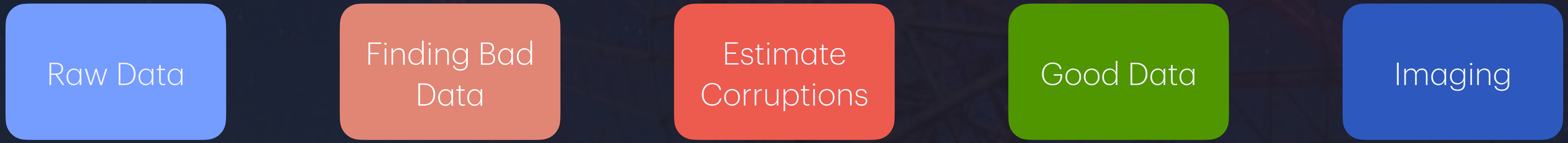
Nimki in Fourier Space



Nimki with an Interferometer



The Road, with Lots of Potholes



- The problem is that, not a single coherent process.
- Different complexities, different experiences.
- Extremely time-taking, due to the inherent nature of repetition.
- But....

NCRA • TIFR

The Pipeline: CHARIZARD

CHARIZARD: Calibration and Highly Automated Radio Imaging with polariZation by Advanced Resource Distribution

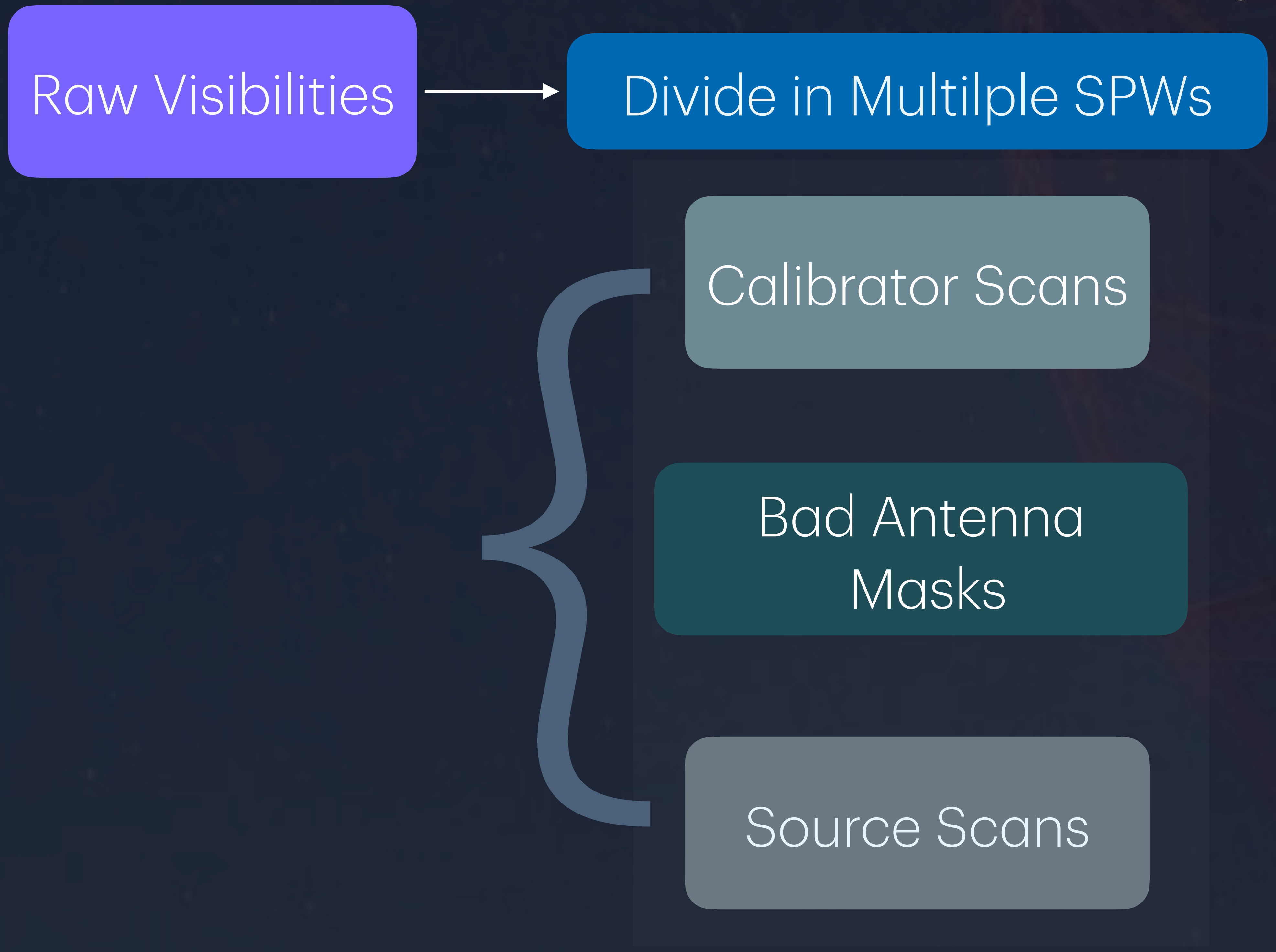
1. A Cross-platform HPC-based imaging pipeline. For now, can work with Slurm and PBS.
2. Not so flexible with algorithms but tries to make use of tested workflow.
3. The idea is to find out jobs which can be run independently.
4. Then run them parallelly. The portions that could not be run. Wait for them.
5. But try to maintain the throughput as much as possible.

[https://github.com/arpan-52/CHARIZARD]

NCRA • TIFR

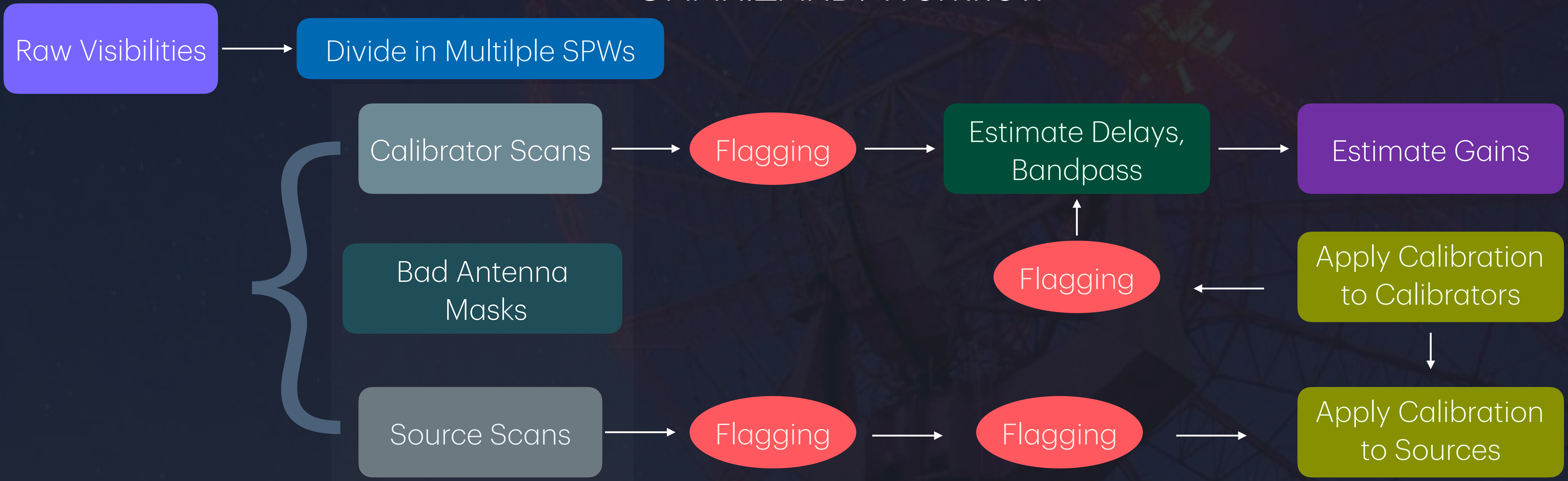
[Pal, A et al. in Prep.]

CHARIZARD: Workflow



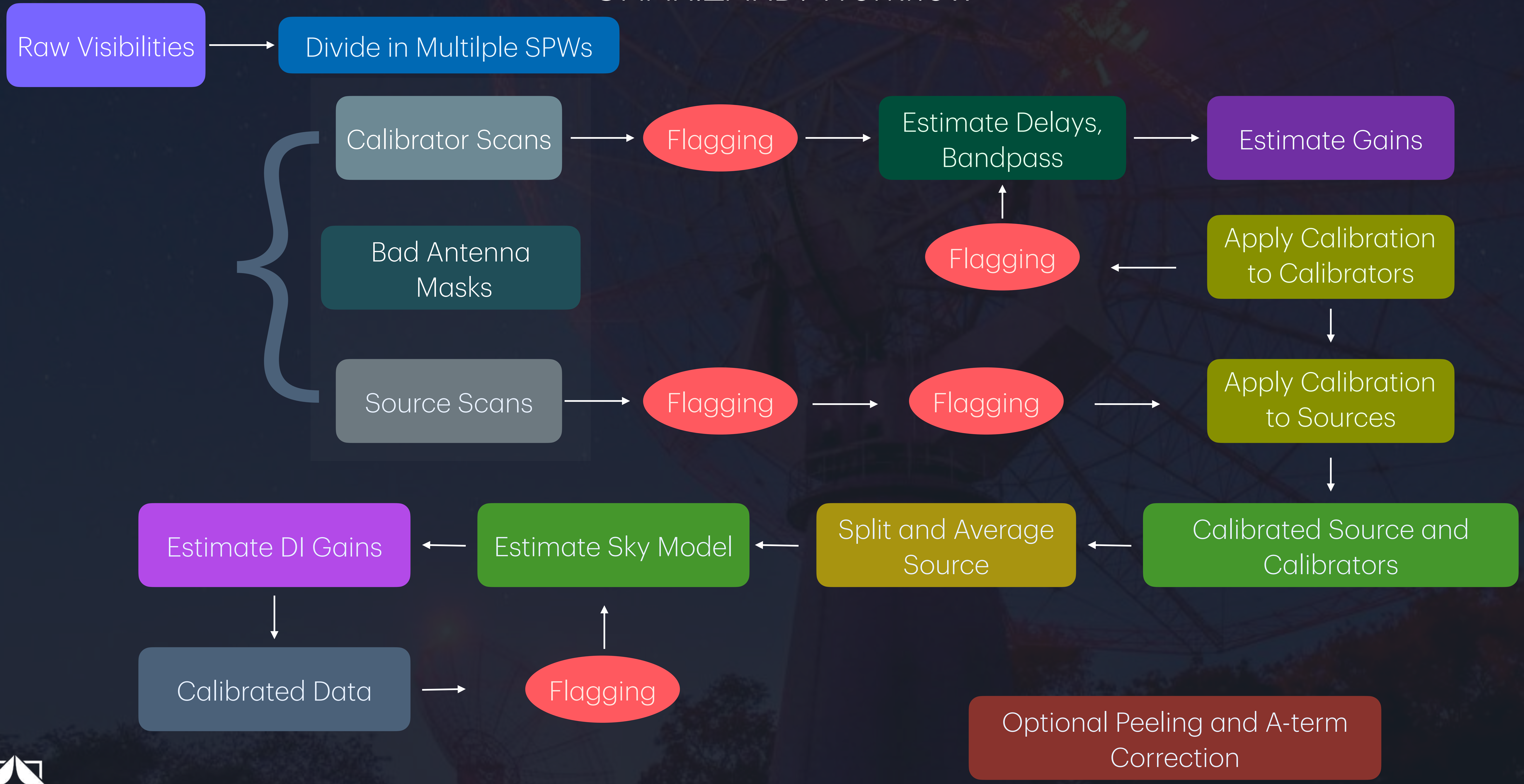
[Pal, A et al. in Prep.] 6

CHARIZARD: Workflow



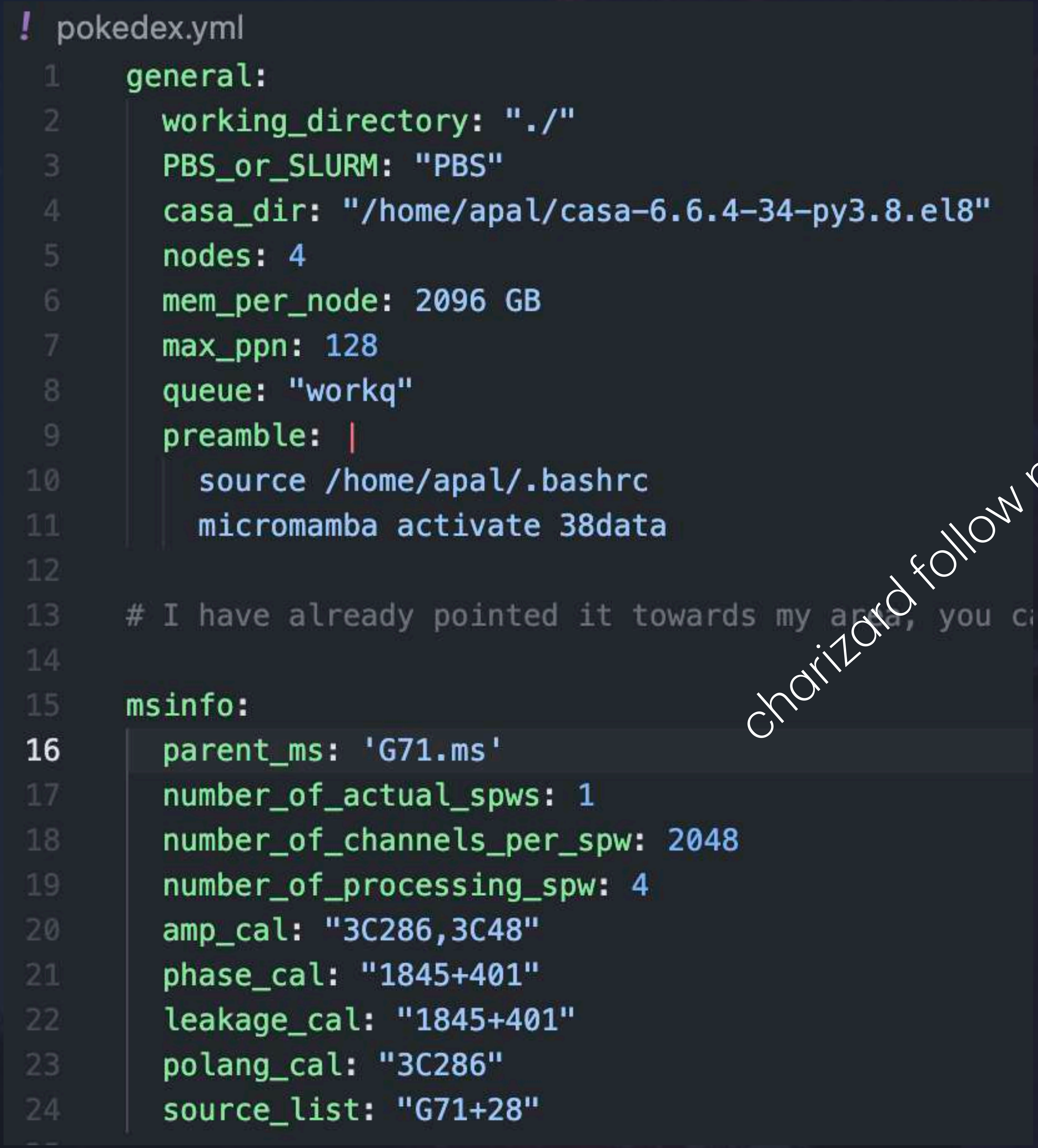
[Pal, A et al. in Prep.]

CHARIZARD: Workflow



[Pal, A et al. in Prep.]

A Simple Config, But More to Come...



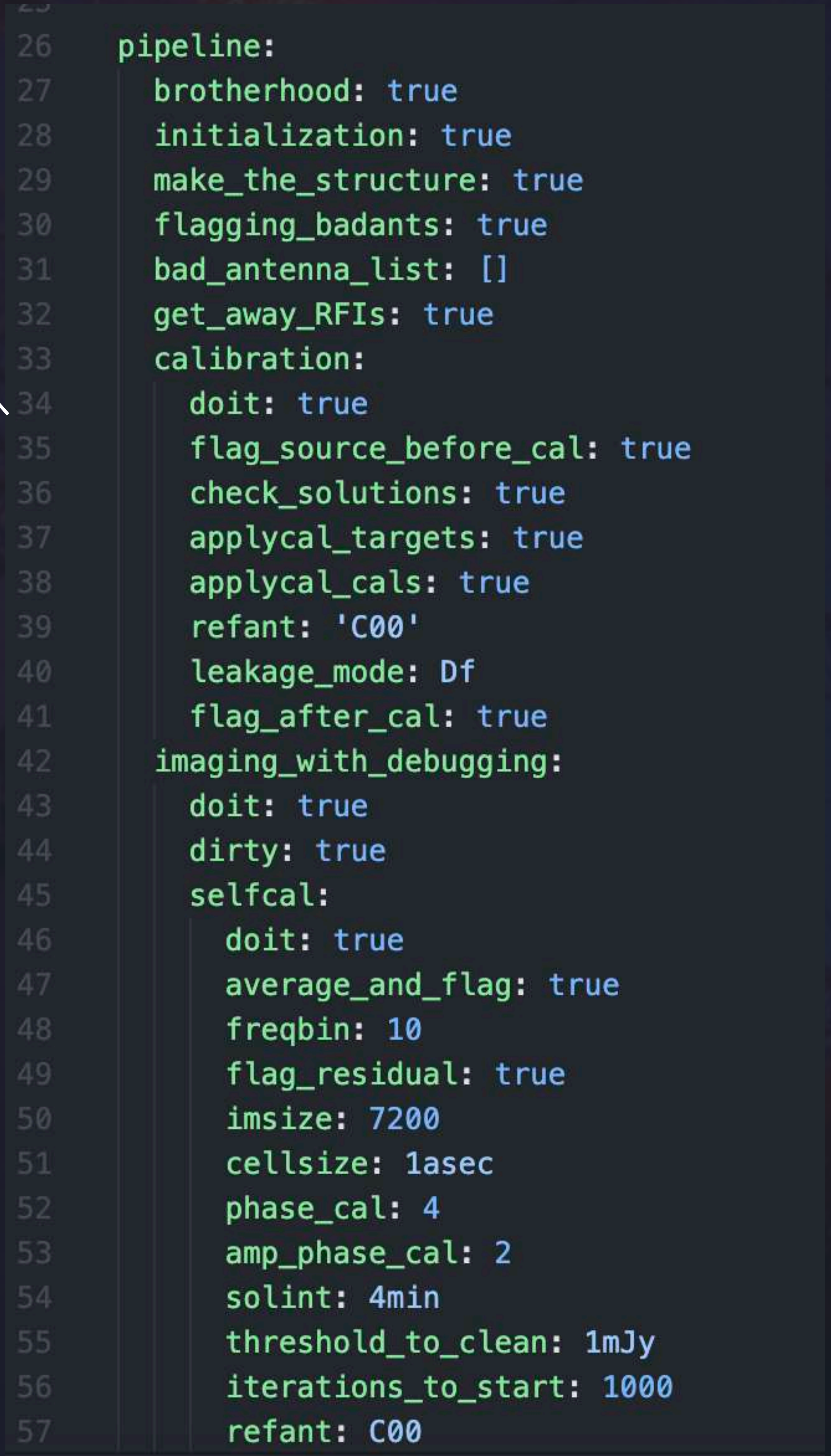
pokedex.yml

```yaml
general:
  working_directory: "./"
  PBS_or_SLURM: "PBS"
  casa_dir: "/home/apal/casa-6.6.4-34-py3.8.el8"
  nodes: 4
  mem_per_node: 2096 GB
  max_ppn: 128
  queue: "workq"
  preamble: |
    source /home/apal/.bashrc
    micromamba activate 38data

# I have already pointed it towards my area, you c

msinfo:
  parent_ms: 'G71.ms'
  number_of_actual_spws: 1
  number_of_channels_per_spw: 2048
  number_of_processing_spw: 4
  amp_cal: "3C286,3C48"
  phase_cal: "1845+401"
  leakage_cal: "1845+401"
  polang_cal: "3C286"
  source_list: "G71+28"
```

charizard follow pokedex.yml



```yaml
pipeline:
  brotherhood: true
  initialization: true
  make_the_structure: true
  flagging_badants: true
  bad_antenna_list: []
  get_away_RFIs: true
  calibration:
    doit: true
    flag_source_before_cal: true
    check_solutions: true
    applycal_targets: true
    applycal_cals: true
    refant: 'C00'
    leakage_mode: Df
    flag_after_cal: true
  imaging_with_debugging:
    doit: true
    dirty: true
    selfcal:
      doit: true
      average_and_flag: true
      freqbin: 10
      flag_residual: true
      imsize: 7200
      cellsize: 1asec
      phase_cal: 4
      amp_phase_cal: 2
      solint: 4min
      threshold_to_clean: 1mJy
      iterations_to_start: 1000
      refant: C00
```

Acceleration in Calibration

Calibration is estimating the corruptions and an easy way to accelerate it is:

- Separate the calibrators scans. Less memory occupancy, easier estimations.
- More number of calibration loops can be tried to look for convergence.
- Makes also flagging quite memory efficient. For version 1.0 we are grouping calibrators, but....
- Polarization calibration is also implemented.
- Leakage and cross-hand phases are solved using CASA. Models are predefined in a dictionary.
- Also the option is there to read the calibration tables and look for outliers.

Adding calibrator models, especially for polarization is straightforward with a dictionary call.

[Pal, A et al. in Prep.]

Acceleration in Flagging

- Now the main show stopper: Flagging (༎ຶ⌑༎ຶ)

  1. Each and every bit of the data has to be checked, statistics have to be calculated and applied.
  2. Typically a calibrator occupies 25-30% of the total data.
  3. Source flagging often is limited by available memories.

- Option 1: Again chunk it into parts, which makes it more unmanageable.
- Option 2: Add sophistication in the flagging algorithms that can handle large data.

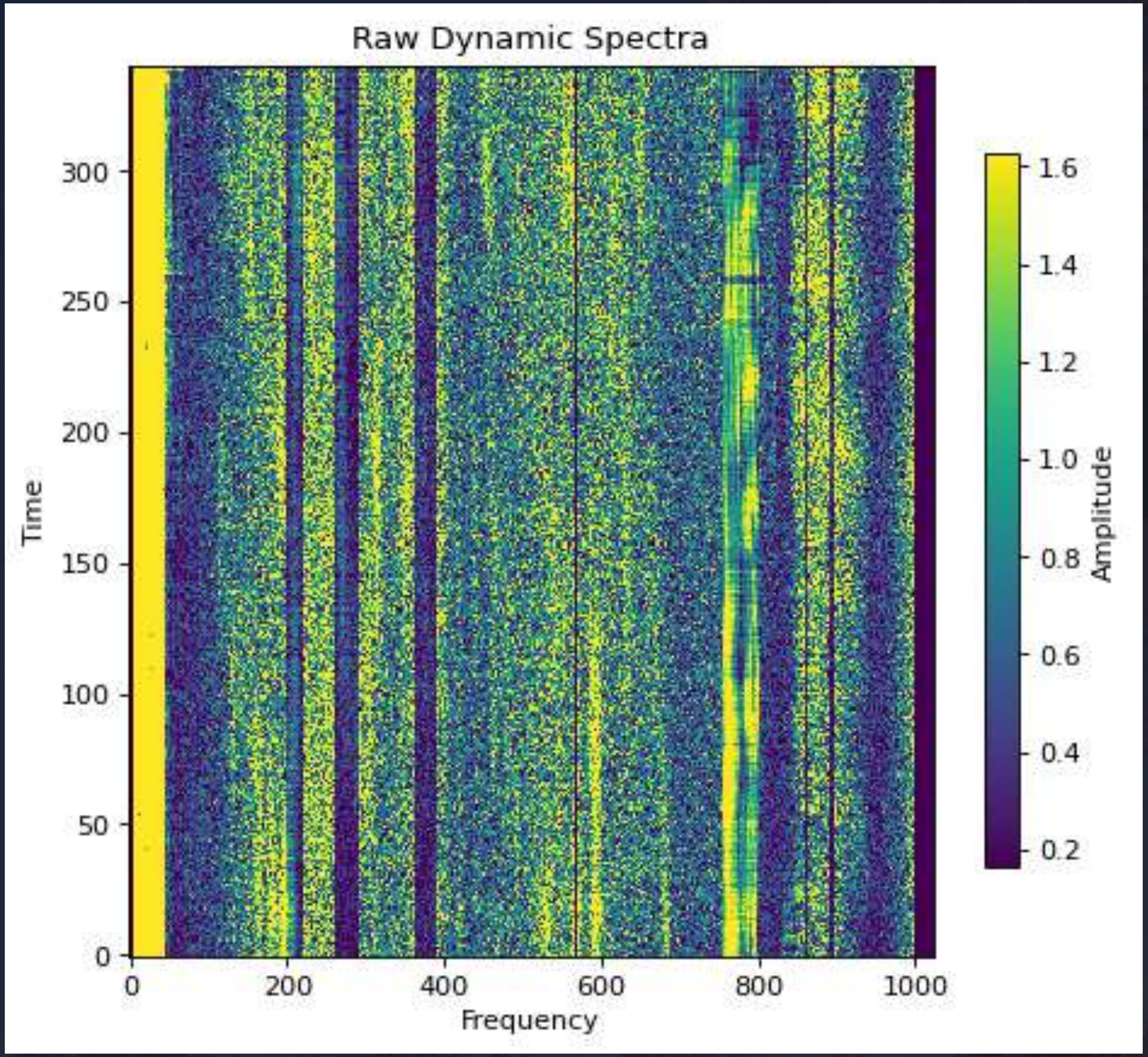
Available algorithms: CASA tfcrop, rflag, AOflagger, tricolour etc.

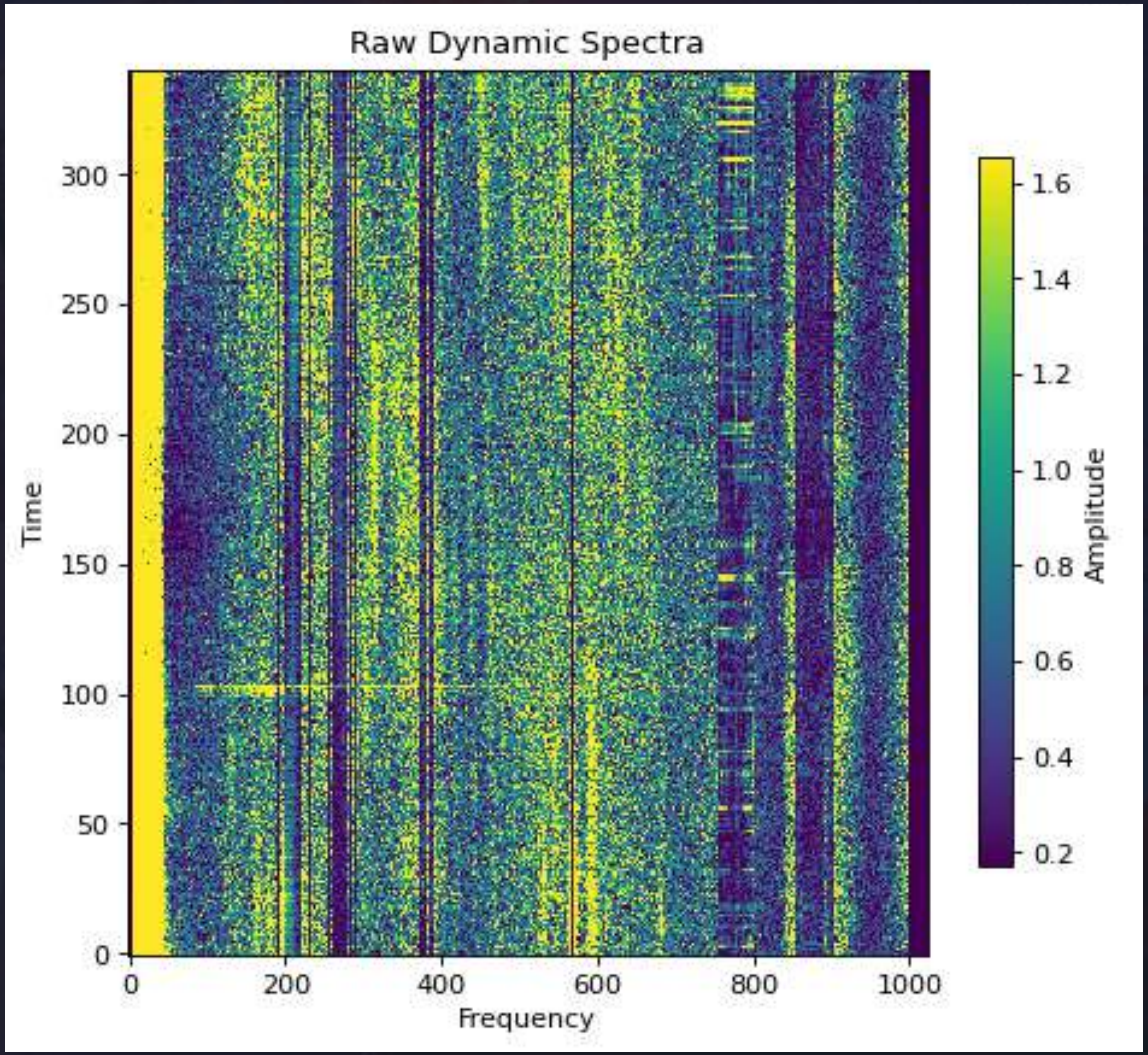
In CHARIZARD, we have 2 new flaggers, looking at the demands.

The idea is simple, do not compromise accuracy, but try to accelerate as much as possible. To start with.....

NCRA • TIFR

Some Examples





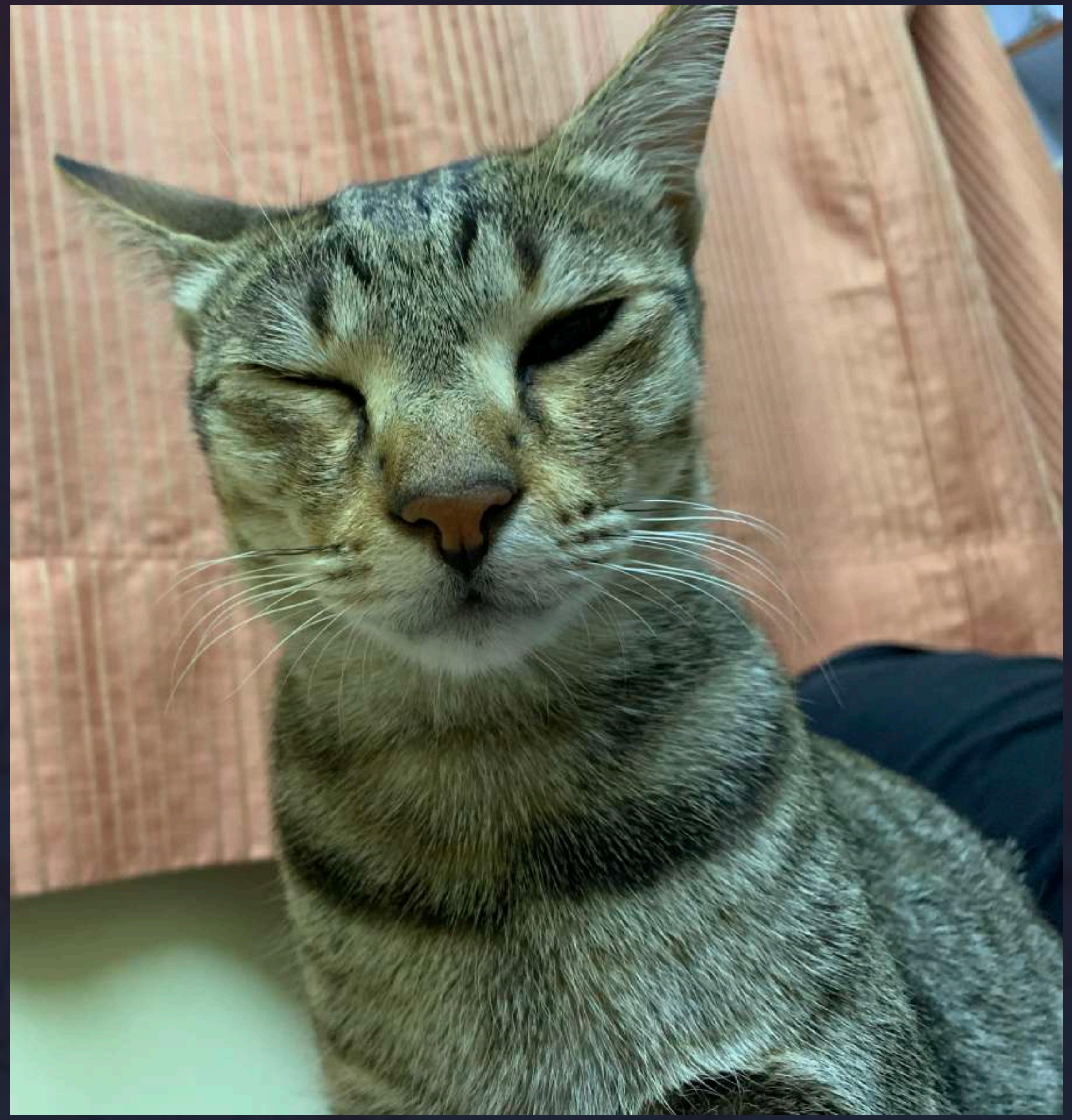
NCRA • TIFR

Acceleration in Flagging

The suite is called CATBOSS. No acronym this time.

For now, 2 flagging algorithms are implemented with different modes.

1. Pooh: Parallelized Optimized Outlier Hunter


Pooh

```
catboss -cat pooh —combinations 1,2,4,8,16 — sigma 5 —rho 1.5 — diagnostic plots —selections —apply-flags {ms}
```

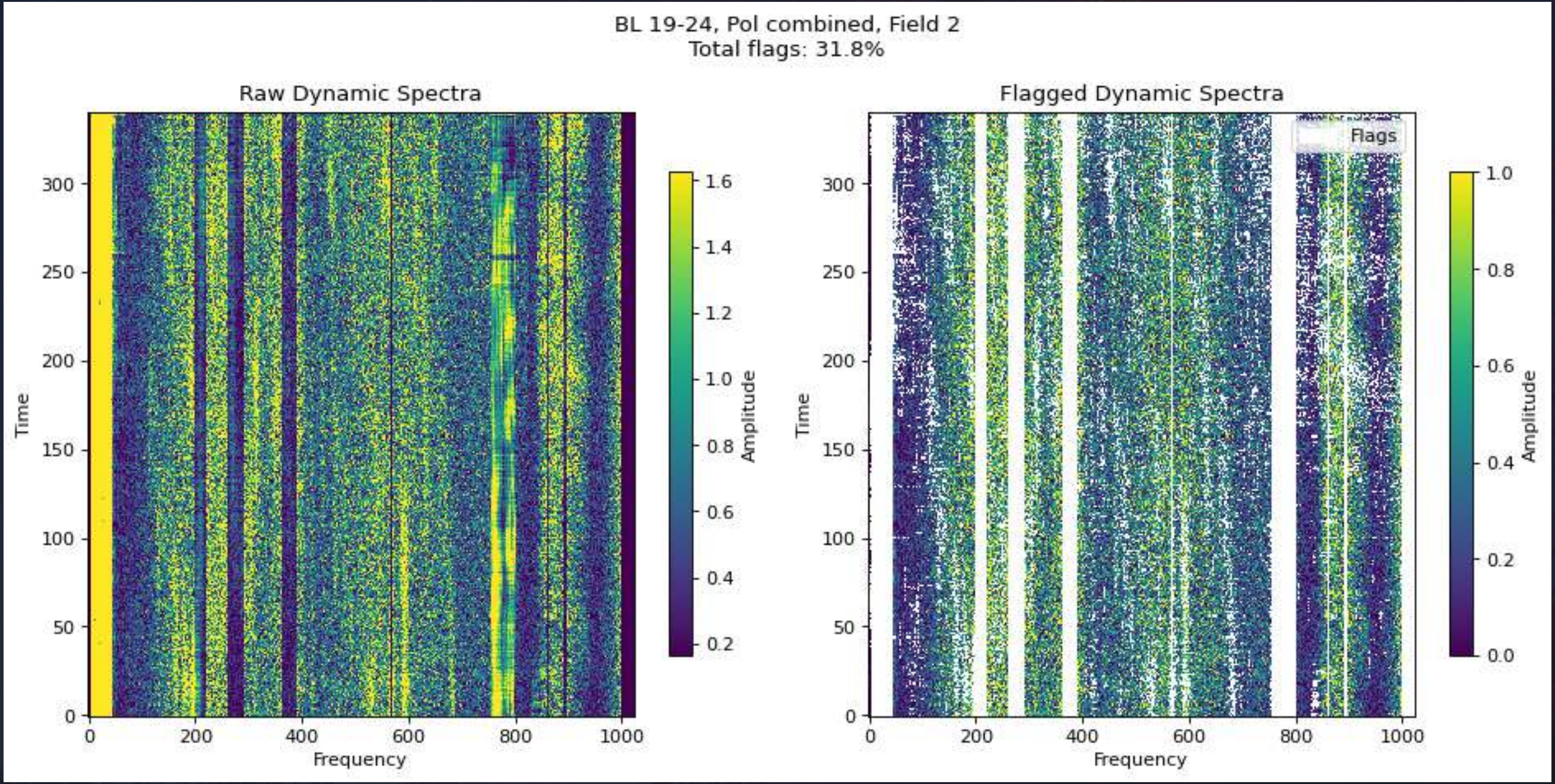
Acceleration in Flagging

The workflow is:

1. First get an idea about the data, read field by field.
2. Estimate bandpass response for each of the baselines and field combinations.
3. A 4D polynomial is fitted in the bandpass, and persistent bad channels are identified.
4. Estimate the thresholds to flag and move everything to GPU.
5. Based on the other unflagged data points, we do a sum-threshold flagging.
6. The whole thing is written to make use of GPU as much as possible. But can be easily further optimized.
7. Performs really well, finding variety of RFI patterns in the spectra.

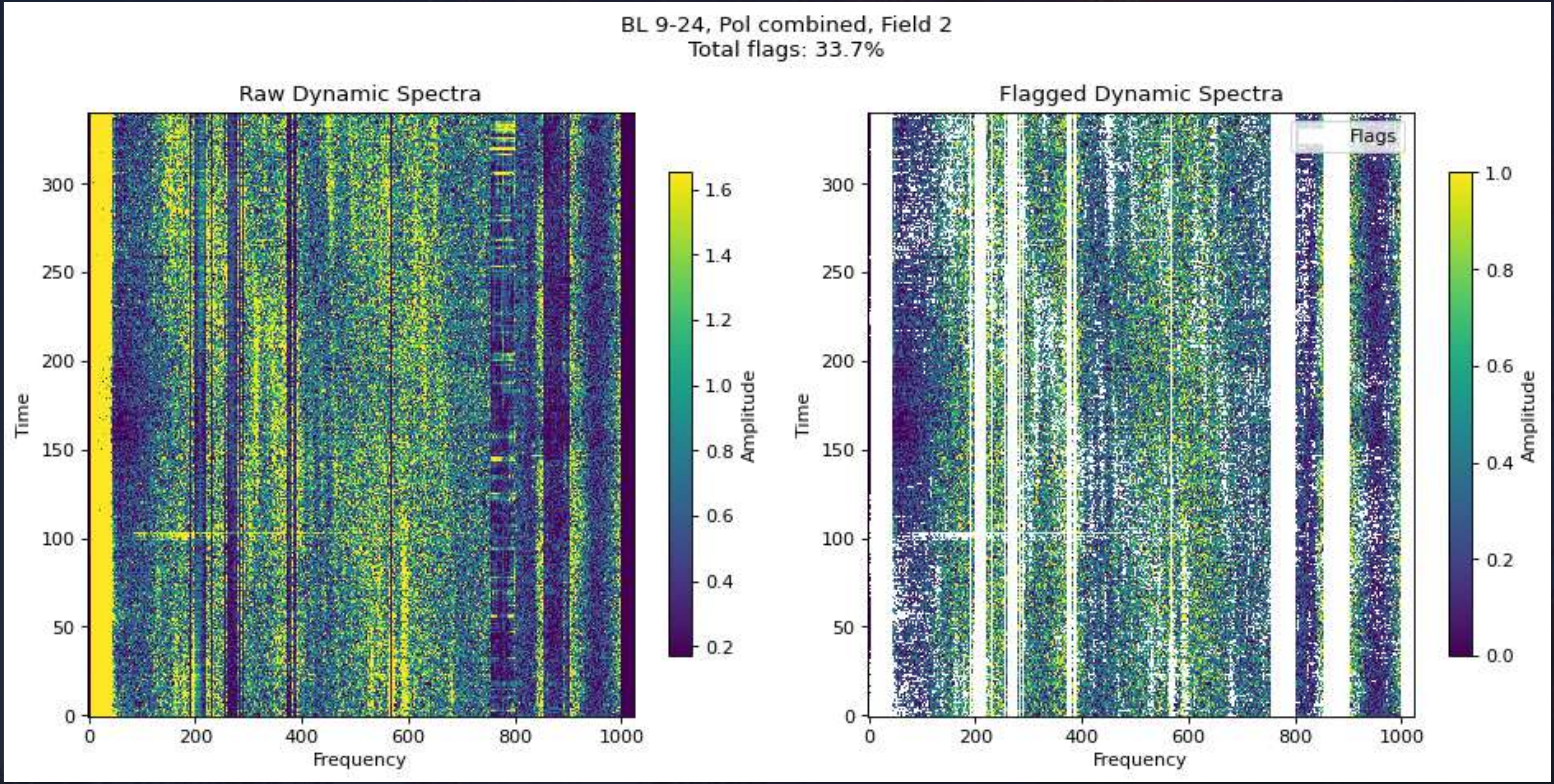
[Pal, A et al. in Prep.]

Acceleration in Flagging



[Pal, A et al. in Prep.]

Acceleration in Flagging



[Pal, A et al. in Prep.]

Acceleration in Flagging

2. NAMI: Nonlinear Automated Monotonous filter for Interference

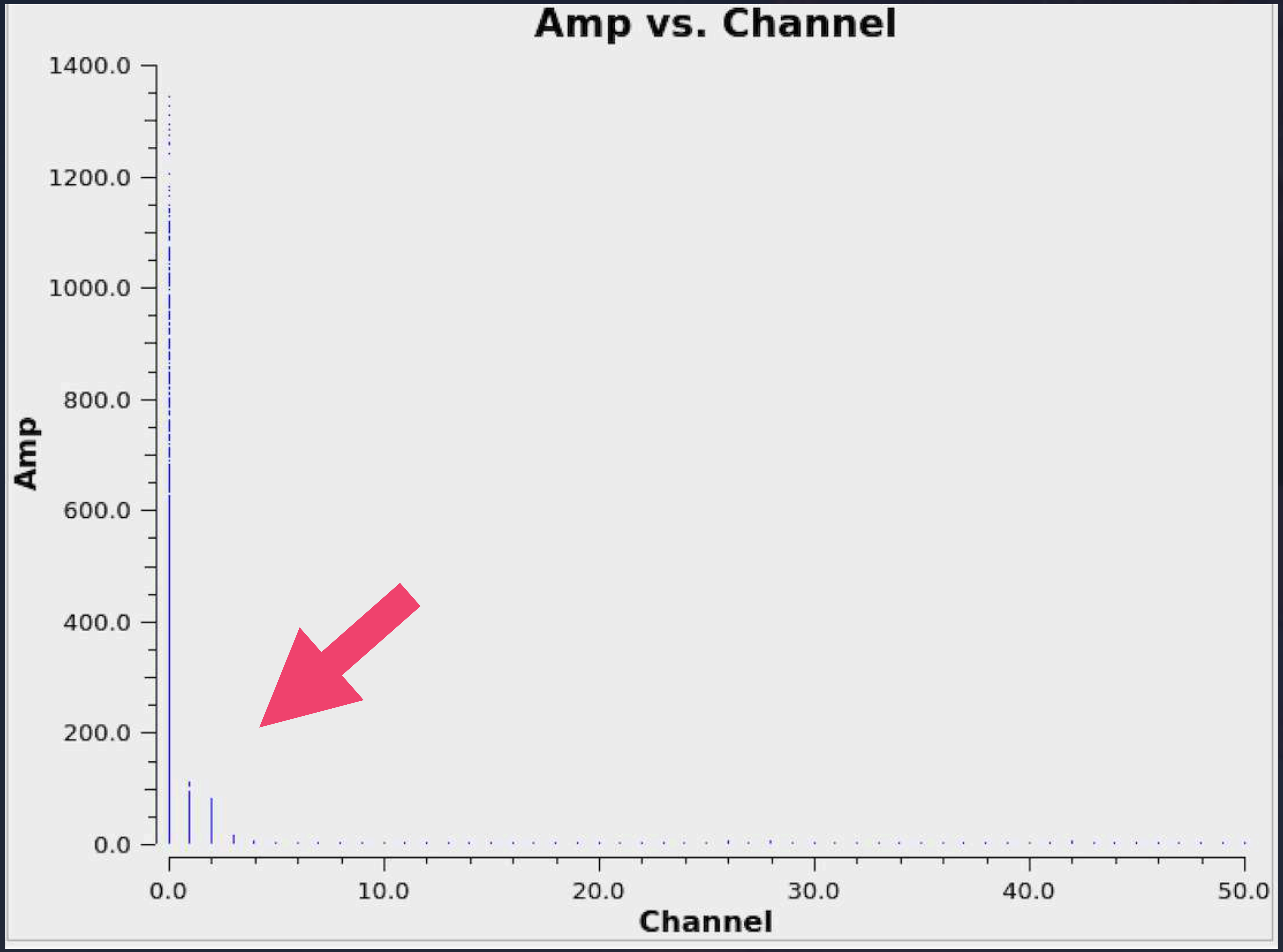
Workflow:

- The idea here is to use the physical information that can be expected from visibility.
- No matter what, properly calibrated visibility should be a monotonous function of the baseline length.
- Point sources are like lines, extended sources are decreasing curves.
- Get a single field, if large enough, chunk it into parts.
- Get every amplitude and put them on a plane where the y-axis is amplitude and the x-axis is uv distance.
- Fit a strictly monotonic function.
- Flag based on residual.
- Performs exceptionally well for calibrated data. However, some issues with maintaining monotonicity.

NCRA • TIFR

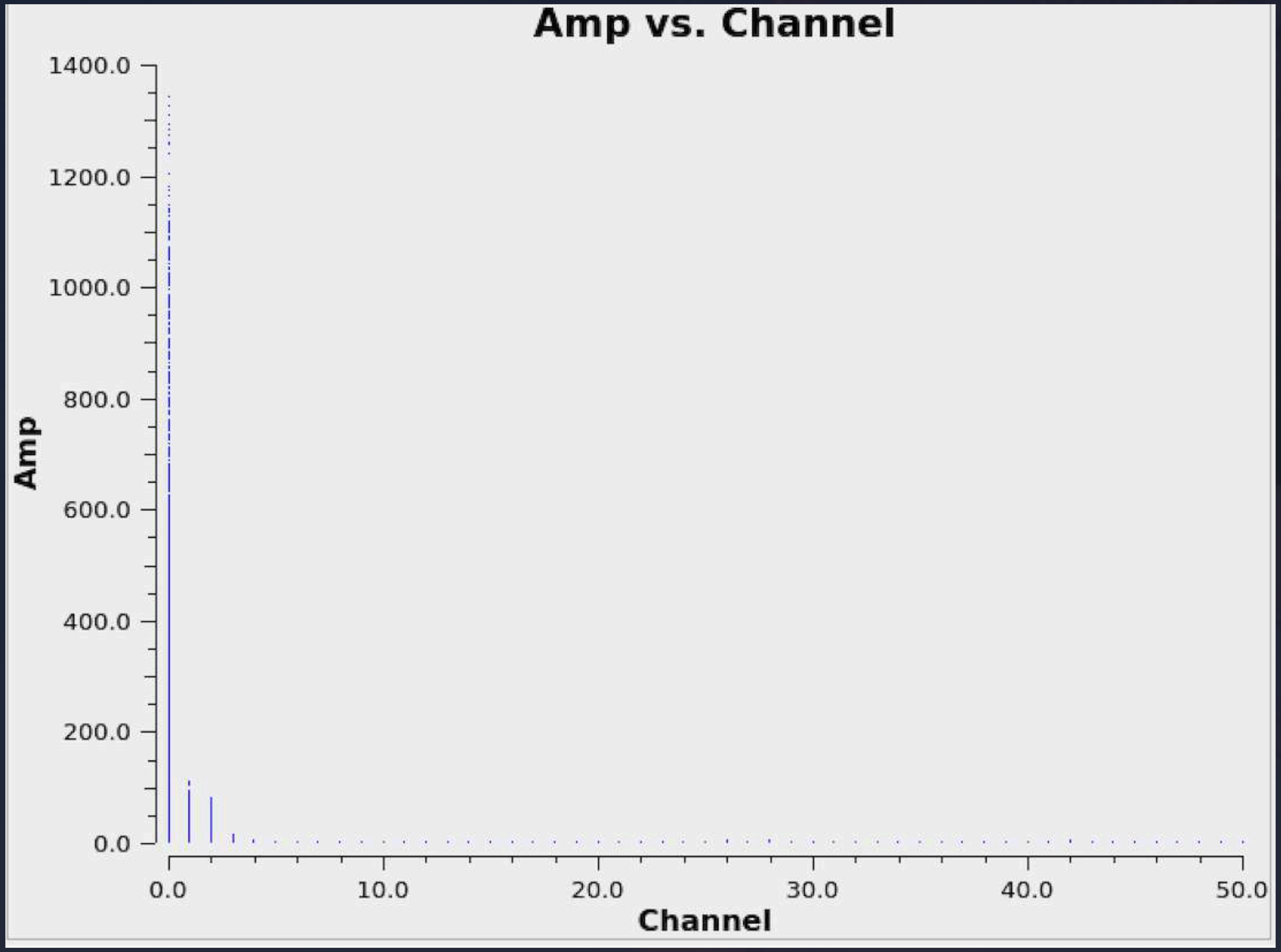
[Pal, A et al. in Prep.]

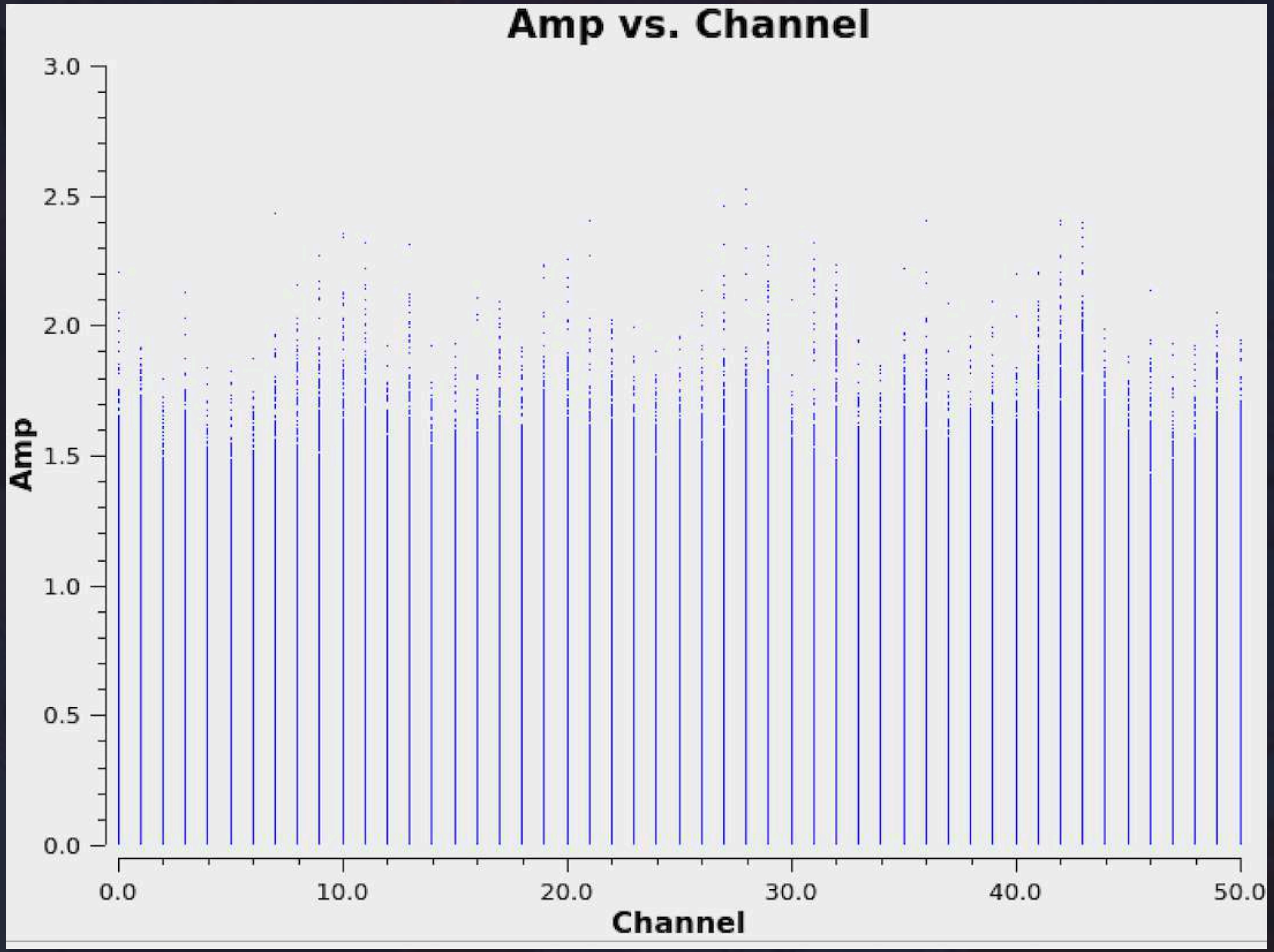
Acceleration in Flagging



[Pal, A et al. in Prep.]

Acceleration in Flagging





[Pal, A et al. in Prep.]

Acceleration in Imaging and Self-Calibration

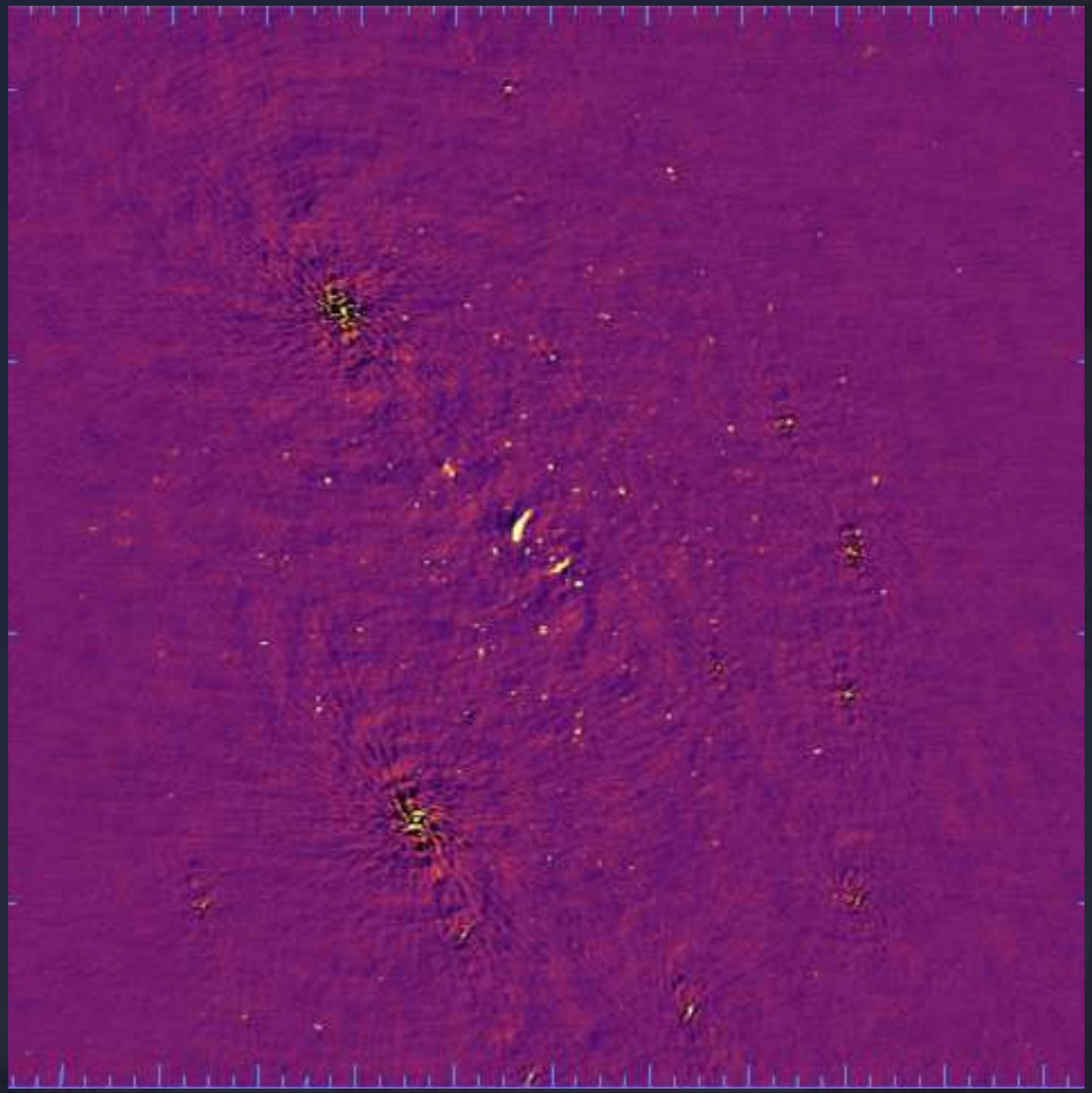
- We do not merge all the subbands. All of them are processed independently.
- We use WSCLEAN for faster W-term corrections and more modularity.
- First a shallow sky model is estimated. Then sources are identified using the pybdsf.
- The mask is further passed for deconvolution in deeper rounds.
- Phase-only solutions are estimated first and applied. Then amplitudes are corrected.
- An experimental mode is there to check the self-calibration solutions and convergence.
- Residual data are flagged with catboss.
- Quite modular in terms of selfcal terms.
- Lastly, an experimental module is also there to test DD algorithms.

NCRA • TIFR

[Pal, A et al. in Prep.]

DD Calibration at uGMRT

- The Pipeline right now, can do a peeling-based DD calibration using cubical, WSCLEAN.
- But .....

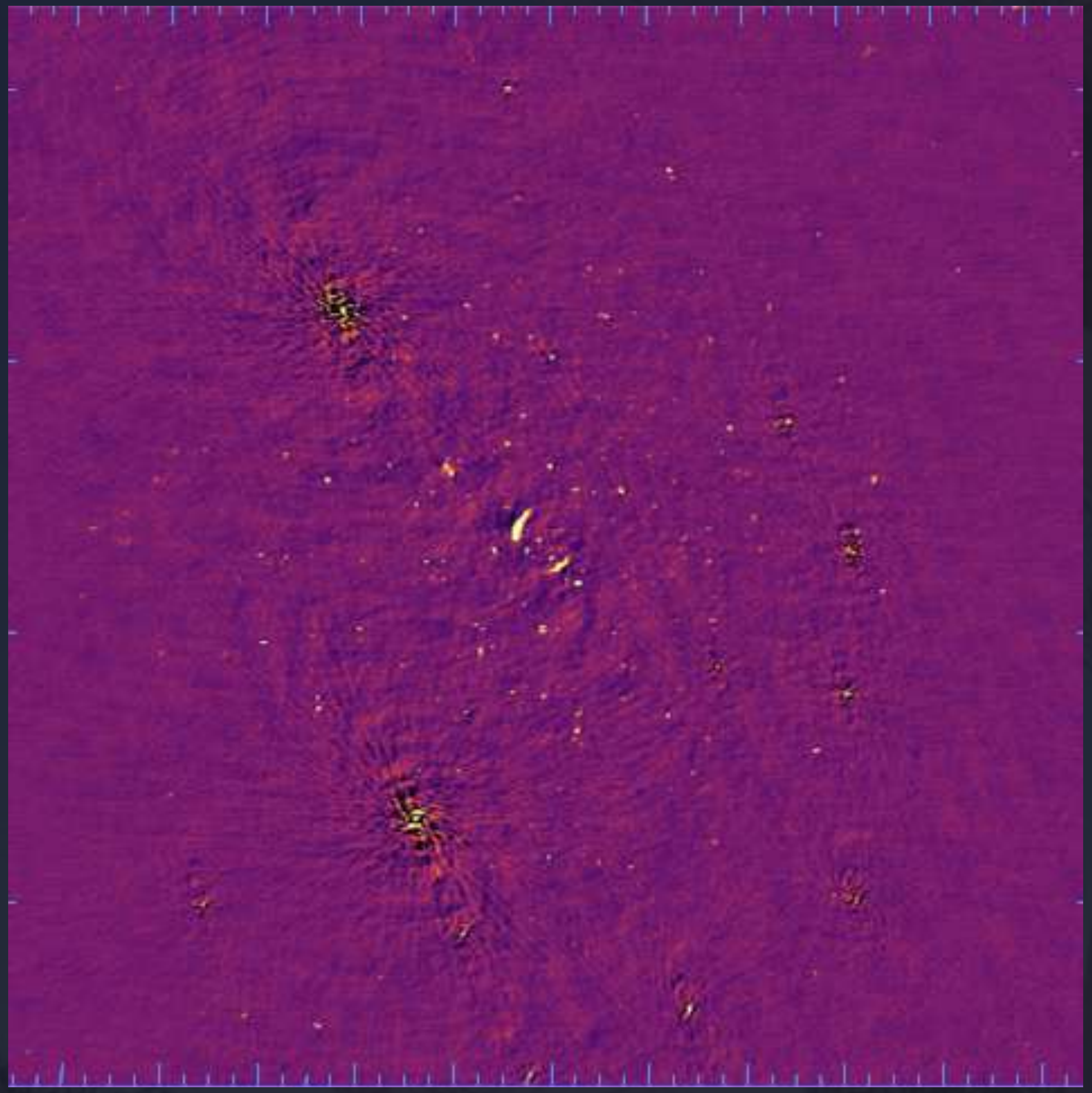

Pre-DD

[Pal, A et al. in Prep.]

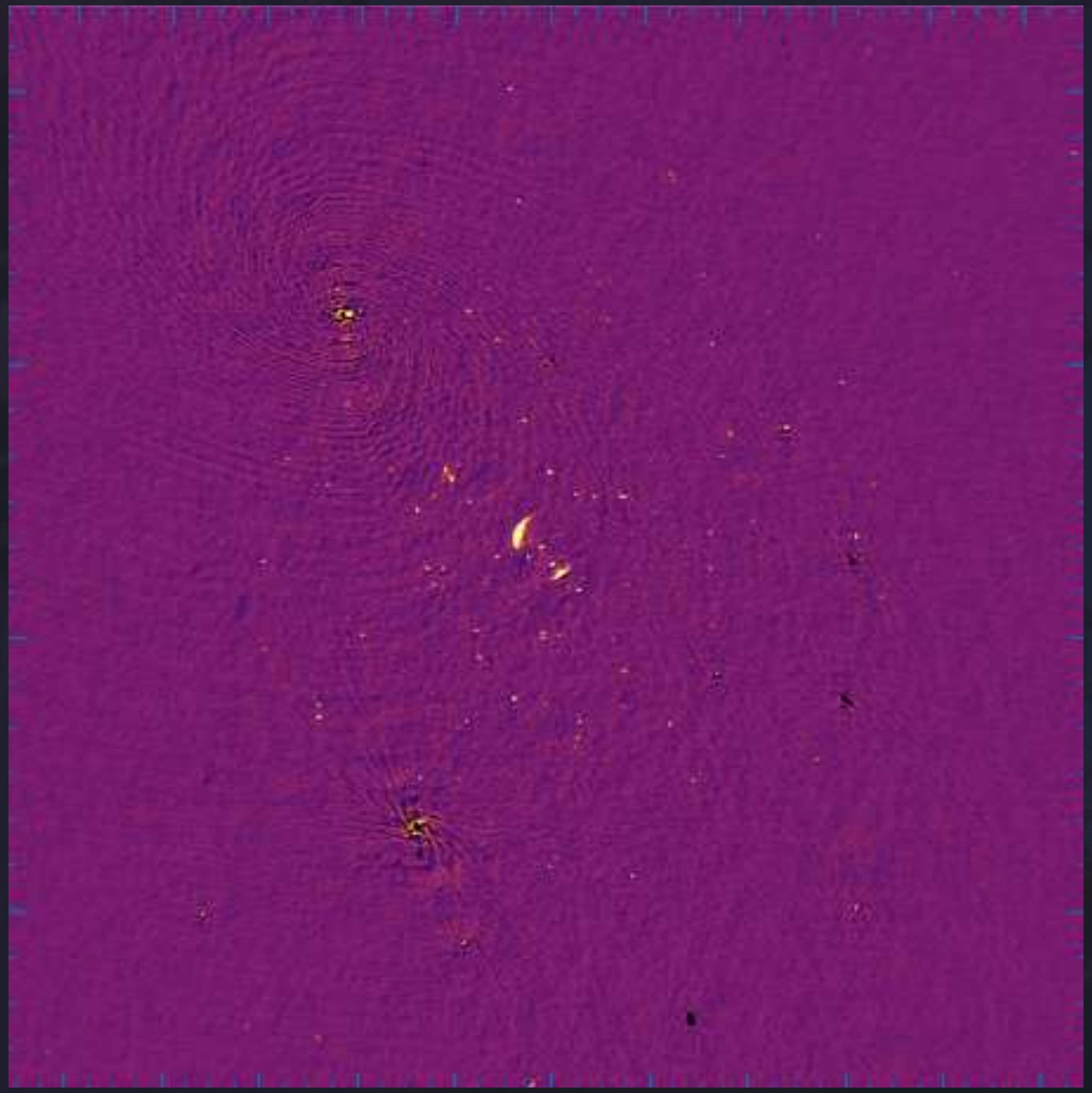
DD Calibration at uGMRT

- The Pipeline right now, can do a peeling-based DD calibration using cubical, WSCLEAN.
- But .....

17 μJy/beam



Pre-DD



Post-DD

12 μJy/beam

[Pal, A et al. in Prep.]

NCRA • TIFR

Some Side Developments and Future Works

- The Pipeline is also tested on VLA data, but a separate VLA pipeline also exists.
- The main difference is that the calibration is done by CASA VLA Pipeline and that is also fully automated.
- In case, if you want to just correct for primary beams for uGMRT and JVLA, use Finalflash.

All of the softwares are installable using pip.

Future plans include:

- Right now, we process uGMRT 4k channels, 5s sub-integration data of 10 hours with a throughput of 0.6-1.
- Testing different calibration algorithms.
- More optimization in terms of speed and accuracy.

NCRA • TIFR

[Pal, A et al. in Prep.]