Laboratoire d'Études Spatiales et d'Instrumentation en Astrophysique

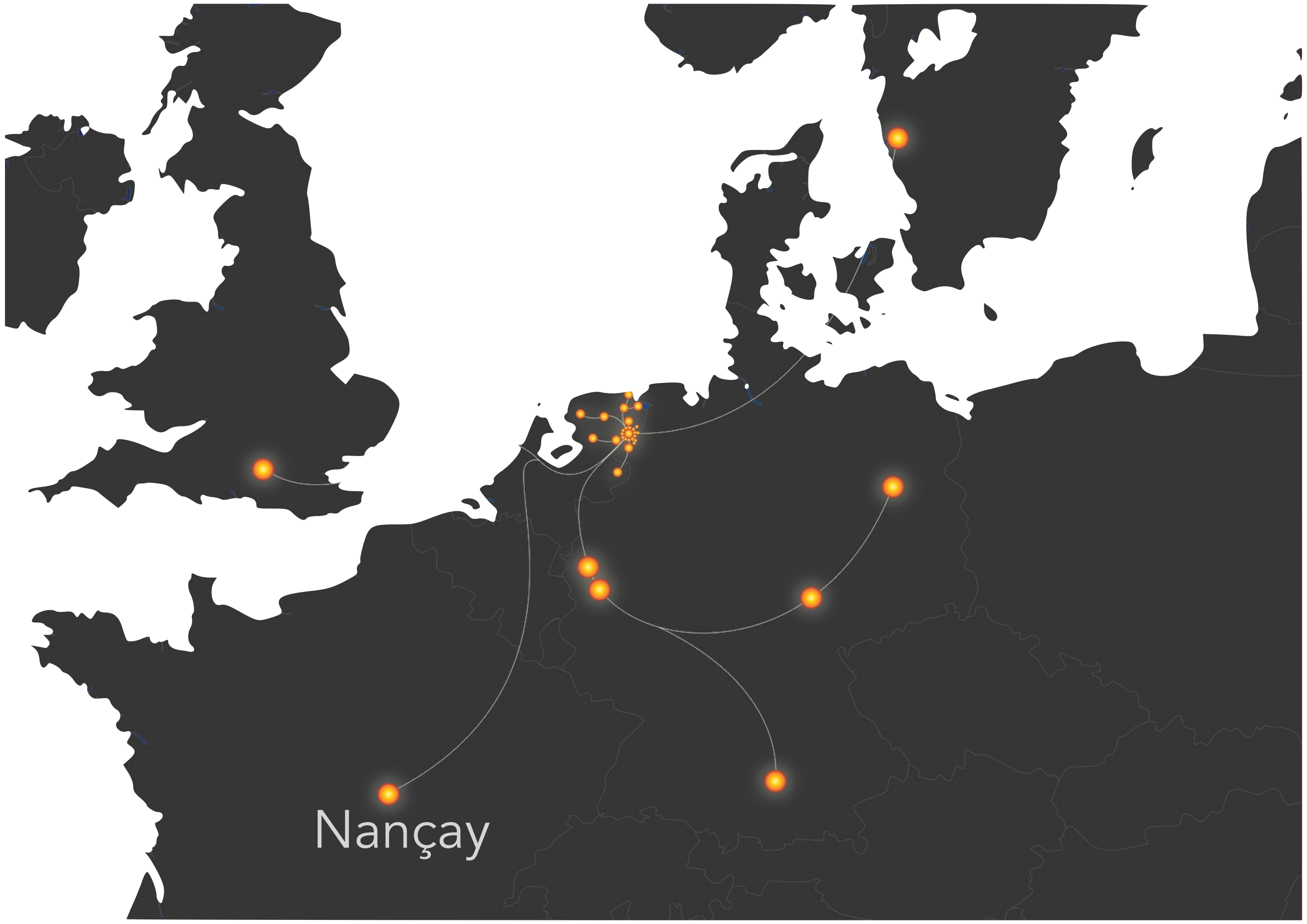
STMOC with NenuFAR

Alan Loh, Lilia Tremou, Baptiste Cecconi

9 March 2020

LOFAR 'SuperStation'

- **LOFAR** (*Low-Frequency Array*), large radio arrays pathfinder: ~50 antenna arrays ('stations') throughout Europe.
  - LBA: 30 - 80 MHz
  - HBA 110 - 250 MHz
- One LOFAR station in *Station de Radioastronomie de Nançay* FR606 with HBA, LBA and LBL data stream inputs.
- Make use of the 96 unused LBL entries to build a *LOFAR SuperStation.*

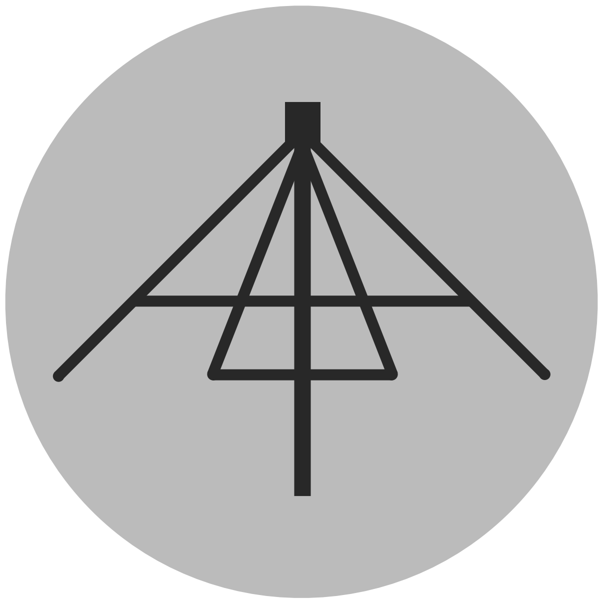


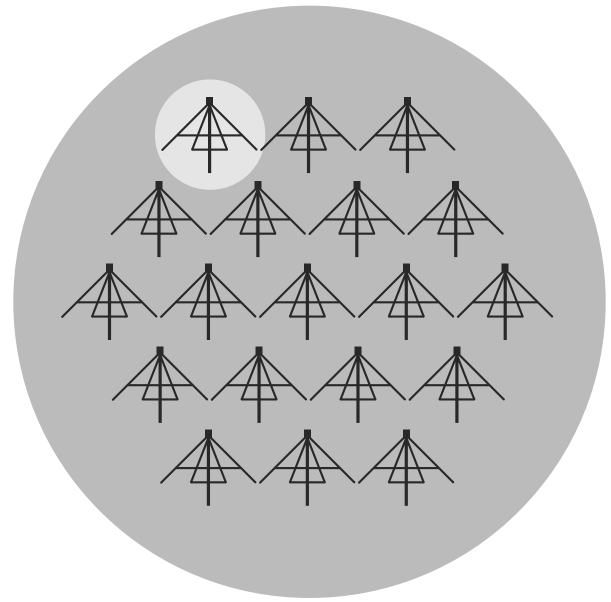
- **NenuFAR** *New Extension in Nançay upgrading LOFAR*:

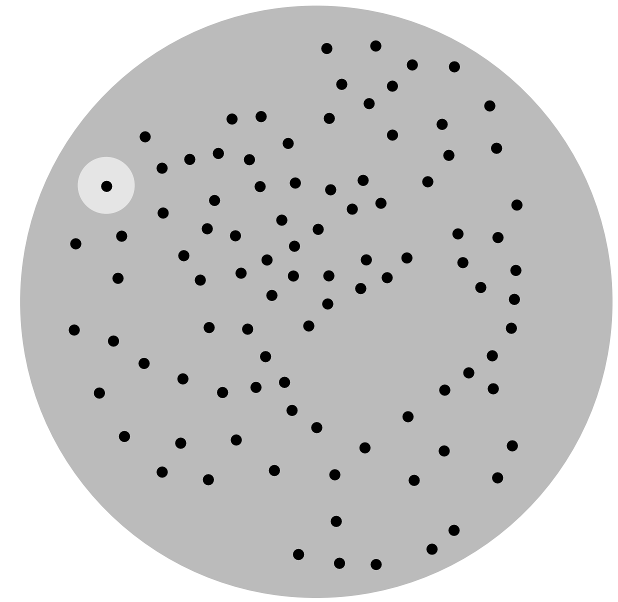
LOFAR SuperStation	**Standalone Imager**	**Standalone Beamformer**
PSF < 1 arcsec	PSF ~10 arcmin	Up to 768 beams
Increase short bsl. sensitivity	96 + 6 distant Mini-Arrays	Pulsar & SETI modes

NenuFAR & LOFAR

NenuFAR







Antenna

1938 LWA-like radiator antennas

Dual-polarizations inverted V shape elements

Low-Noise Amplifier

~All-sky field of view

Broadband response at 10-85 MHz

Mini-Array

Hexagon tile of 19 antennas

Analog beamforming with delay lines

16384 pointable directions on the sky

Beam width: 46° at 15 MHz, 8° at 85 MHz

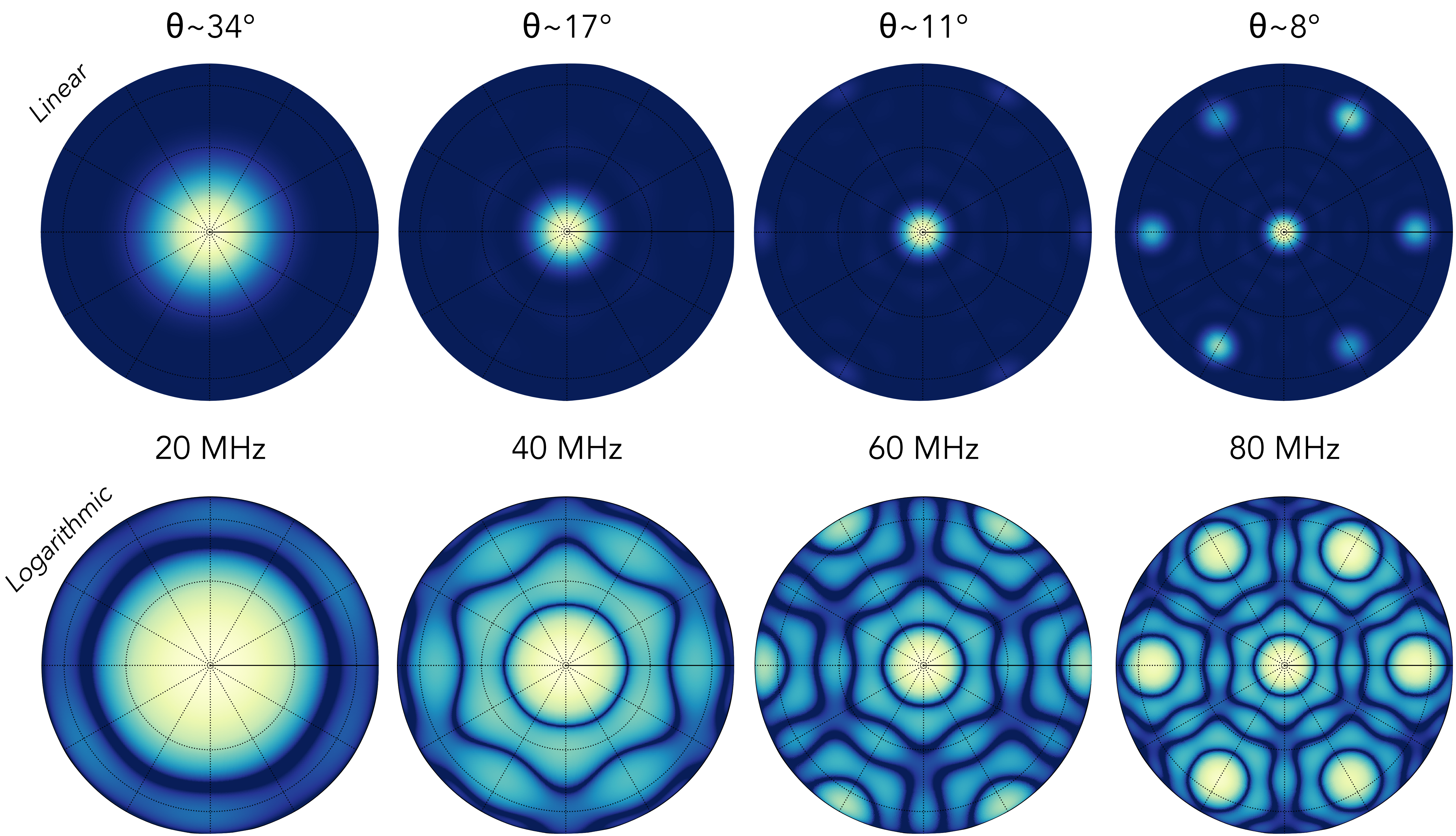
Core

96 mini-arrays (400m core) + 6 remote (up to 3km)

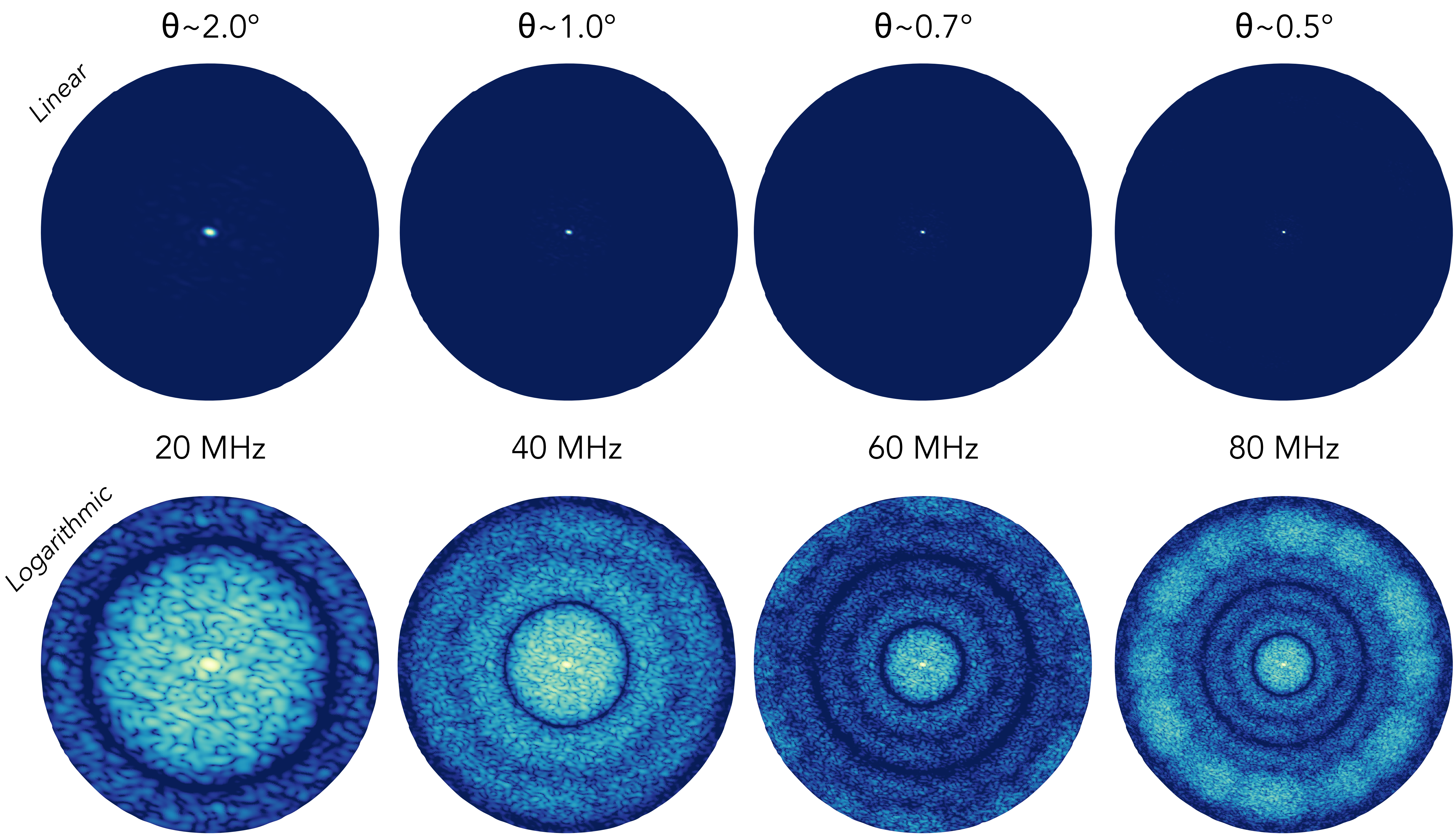
Optimal uv plane coverage for snapshots

Relative MA rotations: dampen grating lobes

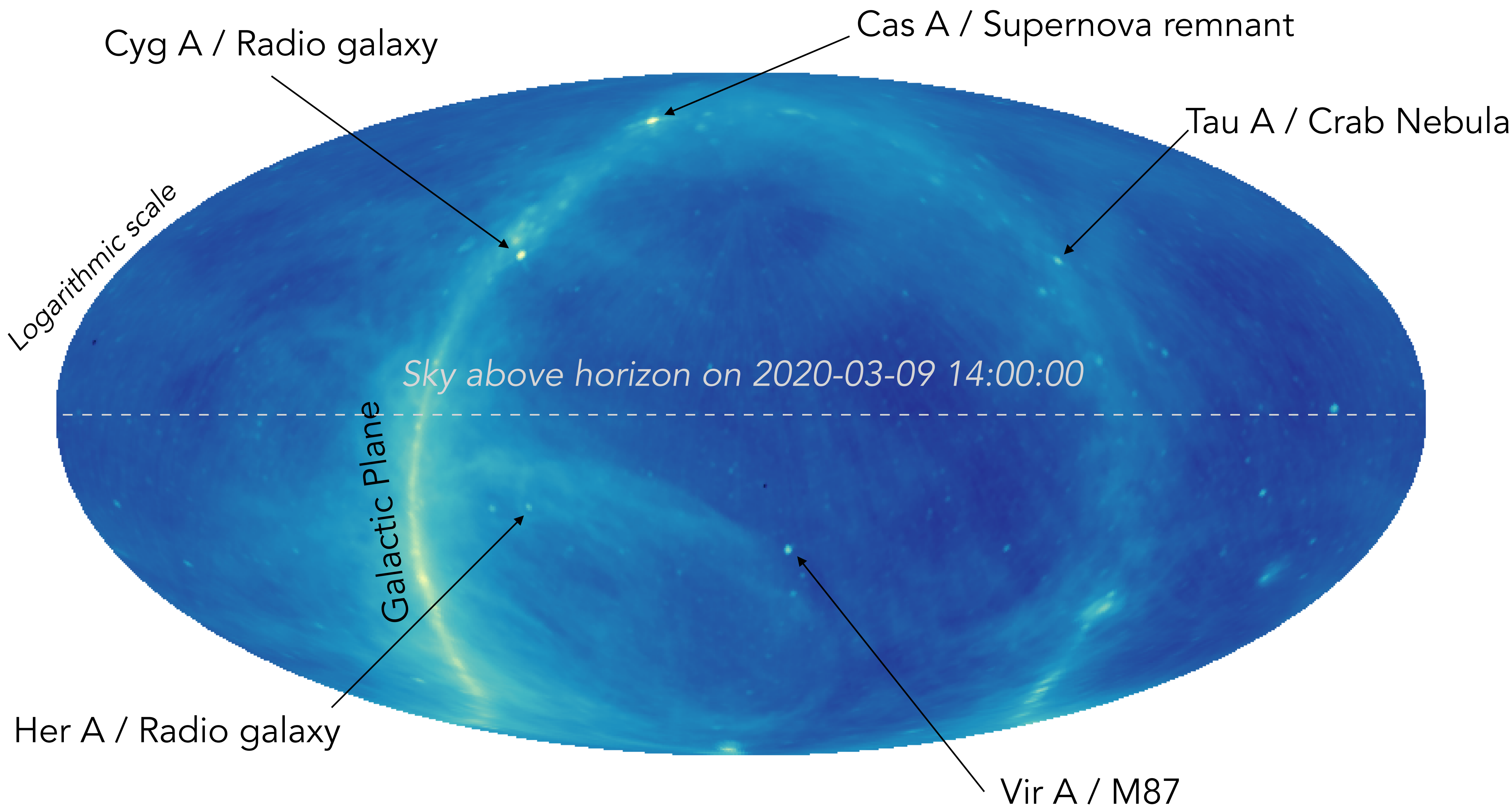
NenuFAR Mini-Array Beam



NenuFAR 56-MAs Beam



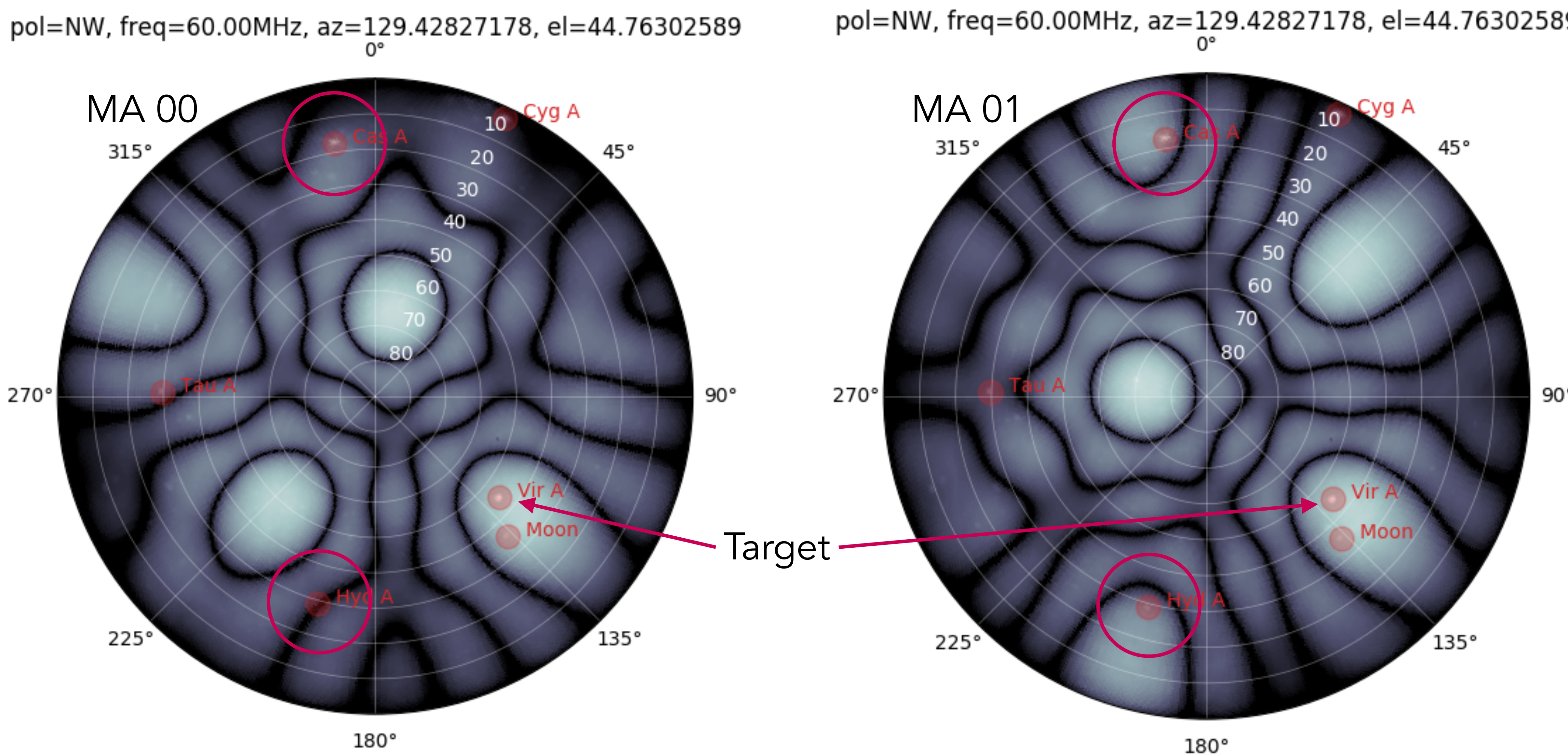
Low-Frequency Sky



GSM Oliveira-Costa et al., 2008

Bright sources in Grating Lobes

Two Mini-Arrays phased at the same position with different rotations:



Signal is summed during beamforming: very hard to disentangle contributions from individual sources.

Effect minimized with the full array summation though.

Using ST(e)MOC for NenuFAR

- Easy access to NenuFAR observations:
  - Query by position/object
  - Query by time (transient event)
  - Query by frequency
- Quickly check if a bright source can affect the data:
  - Within the primary lobe
  - Within the 'grating lobe torus'

NenuFAR observation database

- Many NenuFAR observations and as many configurations…
- Each observation generates a BST file with all necessary **metadata**
- NenuFAR observation database as **TAP service** *(B. Cecconi)*
- Currently gather obs. from 2019-01-01 to 2019-11-05
- Database queries with **pyvo**
- See *http://vogate.obs-nancay.fr/__system__/dc_tables/show/tableinfo/nenufar.bst*

Name	Table Head	Description	Unit	UCD
accref	Product key	Access key for the data	N/A	N/A
owner	Owner	Owner of the data	N/A	N/A
embargo	Embargo ends	Date the data will become/became public	a	N/A
mime	Type	MIME type of the file served	N/A	meta.code.mime
accsize	File size	Size of the data in bytes	byte	VOX:Image_FileSize
dataproduct_type	Dataproduct_type	High level scientific classification of the data product, taken from an enumeration	N/A	meta.id
dataproduct_subtype	Dataproduct_subtype	Data product specific type	N/A	meta.id
calib_level	Calib_level	Amount of data processing that has been applied to the data [Note calib]	N/A	meta.code;obs.calib
obs_collection	Obs_collection	Name of a data collection (e.g., project name) this data belongs to	N/A	meta.id
obs_id	Obs_id	Unique identifier for an observation	N/A	meta.id
obs_title	Obs_title	Free-from title of the data set	N/A	meta.title;obs
obs_publisher_did	Obs_publisher_did	Dataset identifier assigned by the publisher.	N/A	meta.ref.uri;meta.curation
obs_creator_did	Obs_creator_did	Dataset identifier assigned by the creator.	N/A	meta.id
access_url	Access_url	The URL at which to obtain the data set.	N/A	meta.ref.url
access_format	Access_format	MIME type of the resource at access_url	N/A	meta.code.mime
access_estsize	Access_estsize	Estimated size of data product	kbyte	phys.size;meta.file
target_name	Target_name	Object a targeted observation targeted	N/A	meta.id;src
target_class	Target_class	Class of the target object (star, QSO, ...)	N/A	src.class
s_ra	S_ra	RA of (center of) observation, ICRS	deg	pos.eq.ra
s_dec	S_dec	Dec of (center of) observation, ICRS	deg	pos.eq.dec
s_fov	S_fov	Approximate spatial extent for the region covered by the observation	deg	phys.angSize;instr.fov
s_region	S_region	Region covered by the observation, as a polygon	N/A	pos.outline;obs.field
s_resolution	S_resolution	Best spatial resolution within the data set	arcsec	pos.angResolution
t_min	T_min	Lower bound of times represented in the data set, as MJD	d	time.start;obs.exposure
t_max	T_max	Upper bound of times represented in the data set, as MJD	d	time.end;obs.exposure
t_exptime	T_exptime	Total exposure time	s	time.duration;obs.exposure
t_resolution	T_resolution	Minimal significant time interval along the time axis	s	time.resolution
em_min	Em_min	Minimal wavelength represented within the data set	m	em.wl;stat.min
em_max	Em_max	Maximal wavelength represented within the data set	m	em.wl;stat.max
em_res_power	Em_res_power	Spectral resolving power delta lambda/lamda	N/A	spect.resolution
o_ucd	O_ucd	UCD for the product's observable	N/A	meta.ucd
pol_states	Pol_states	List of polarization states in the data set	N/A	meta.code;phys.polarization
facility_name	Facility_name	Name of the facility at which data was taken	N/A	meta.id;instr.tel
instrument_name	Instrument_name	Name of the instrument that produced the data	N/A	meta.id;instr
s_xel1	S_xel1	Number of elements (typically pixels) along the first spatial axis.	N/A	meta.number
s_xel2	S_xel2	Number of elements (typically pixels) along the second spatial axis.	N/A	meta.number
t_xel	T_xel	Number of elements (typically pixels) along the time axis.	N/A	meta.number
em_xel	Em_xel	Number of elements (typically pixels) along the spectral axis.	N/A	meta.number
pol_xel	Pol_xel	Number of elements (typically pixels) along the polarization axis.	N/A	meta.number
s_pixel_scale	S_pixel_scale	Sampling period in world coordinate units along the spatial axis	arcsec	phys.angSize;instr.pixel
em_ucd	Em_ucd	Nature of the product's spectral axis	N/A	meta.ucd
preview	Preview	URL of a preview (low-resolution, quick-to-retrieve representation) of the data.	N/A	meta.ref.url;datalink.preview
source_table	Source_table	Name of a TAP-queriable table this data originates from. This source table usually provides more information on the the data than what is given in obscore. See the TAP_SCHEMA of the originating TAP server for details.	N/A	meta.id;meta.table

STMOC

- **Space Time HEALPix Multi-Order Coverage map**
- Developed at CDS *(Matthieu Baumann, Thomas Boch et al.)*
- HEALPix sky tessellation
- **IVOA Standard, interoperability**
- Fast comparison mechanism between coverage maps
  - Union
  - Intersection
  - Equality
- Find available dataset for a given position (and time, e.g. for transient search)
- Python package: *https://github.com/cds-astro/mocpy*

MOC from NenuFAR database

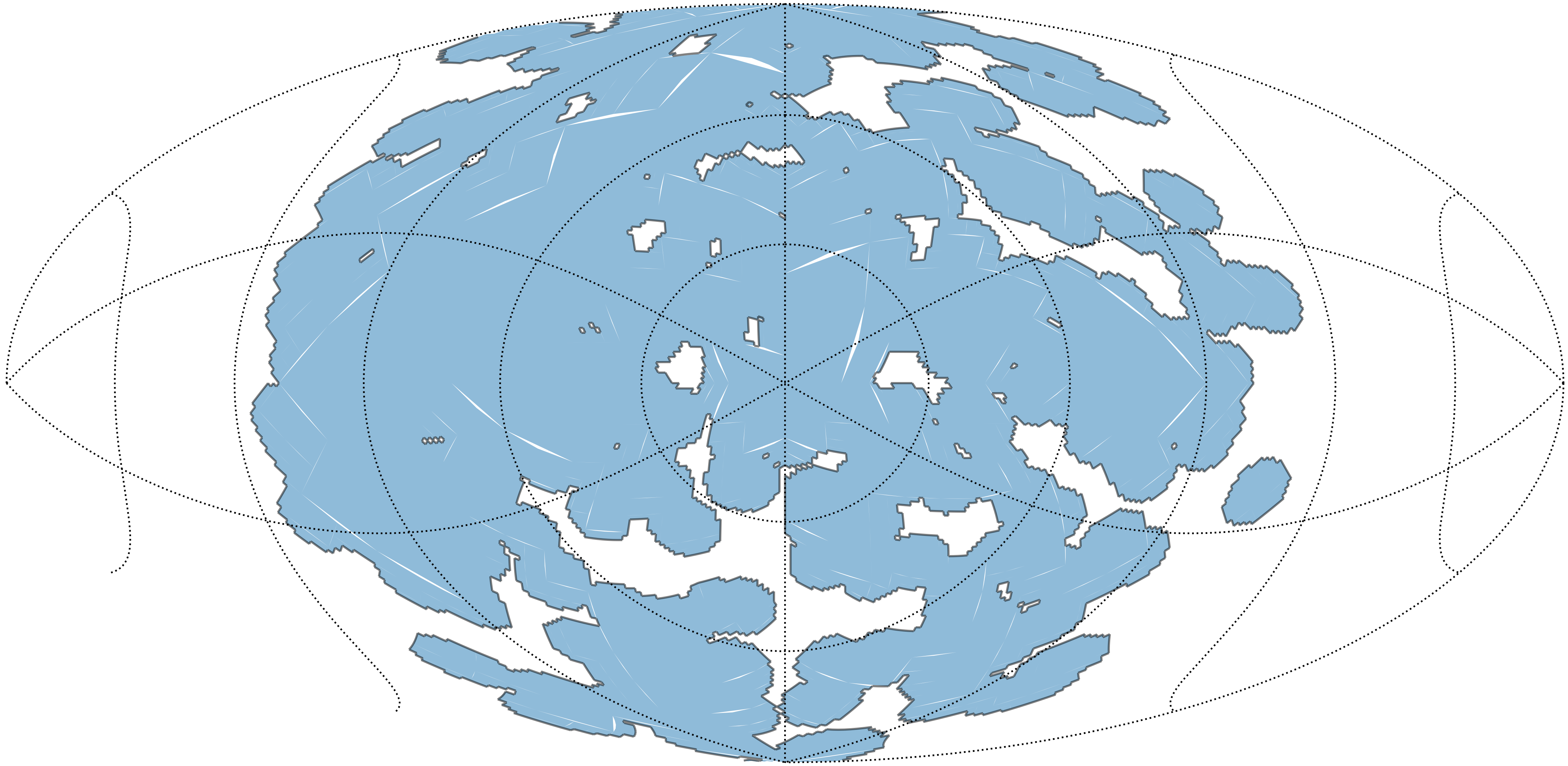
```
from pyvo.dal import TAPService
from cdshealpix import cone_search
from mocpy import MOC
import astropy.units as u
import numpy as np

NANCAY_TAP = 'http://vogate.obs-nancay.fr/tap'

def query_bst():
    service = TAPService(NANCAY_TAP)
    prop_to_return = 'target_name, obs_creator_did, s_ra, s_dec, t_min, t_max, em_min, em_max'
    search_term = '*'
       query = "select {} from nenufar.bst".format(
        prop_to_return
    )
    result = service.search(query)
    return result.to_table()


def make_moc_from_cones(table, ds, freq=30):
    for i in range(len(table)):
        pix, depth, fully_covered_flags = cone_search(
            lon=table['s_ra'][i] * u.deg,
            lat=table['s_dec'][i] * u.deg,
            radius=ma_resol(freq=freq) / 2,
            depth=sresol2order(ds),
            flat=True
        )
        if 'total_pix' in locals():
            total_pix = np.hstack((total_pix, pix))
            total_flags = np.hstack((total_flags, fully_covered_flags))
        else:
            total_pix = pix
            total_flags = fully_covered_flags
    return MOC.from_healpix_cells(total_pix, np.ones_like(total_pix)*depth[0], total_flags)
```

TAP-query the NenuFAR BST metadata database

Build the total MOC made of all observation Primary Beam coverages at a given frequency

Maybe better with MOC.union()...

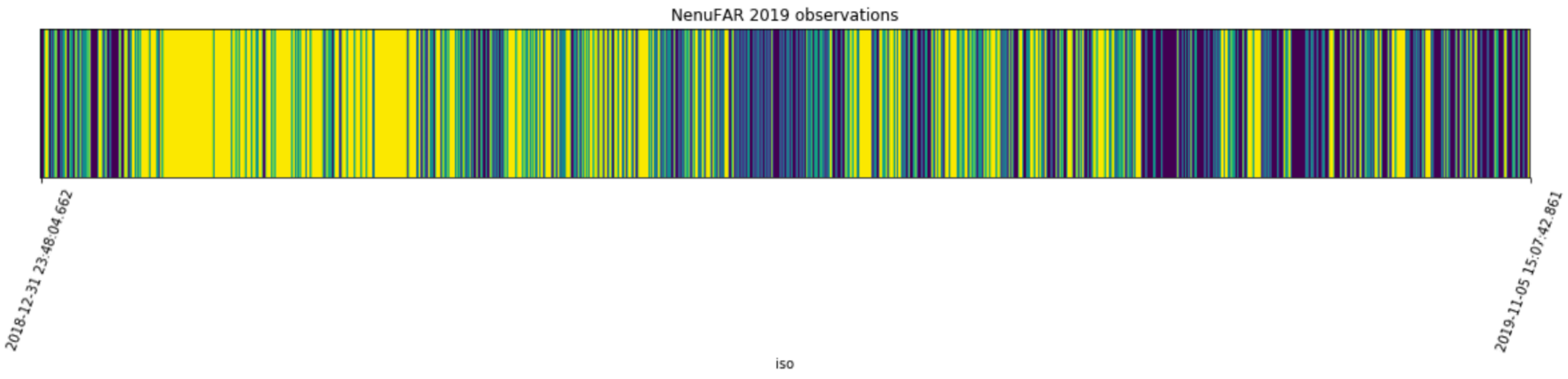
MOC from NenuFAR database



Northern sky coverage of all 2019 observations' primary beam computed at 60 MHz

TMOC from NenuFAR database

```
from mocpy import TimeMOC

time_moc = TimeMOC.from_times(Time(table['t_min'], format='mjd', scale='tdb'))

time_moc.plot(title='NenuFAR 2019 observations', figsize=(20, 7))
```



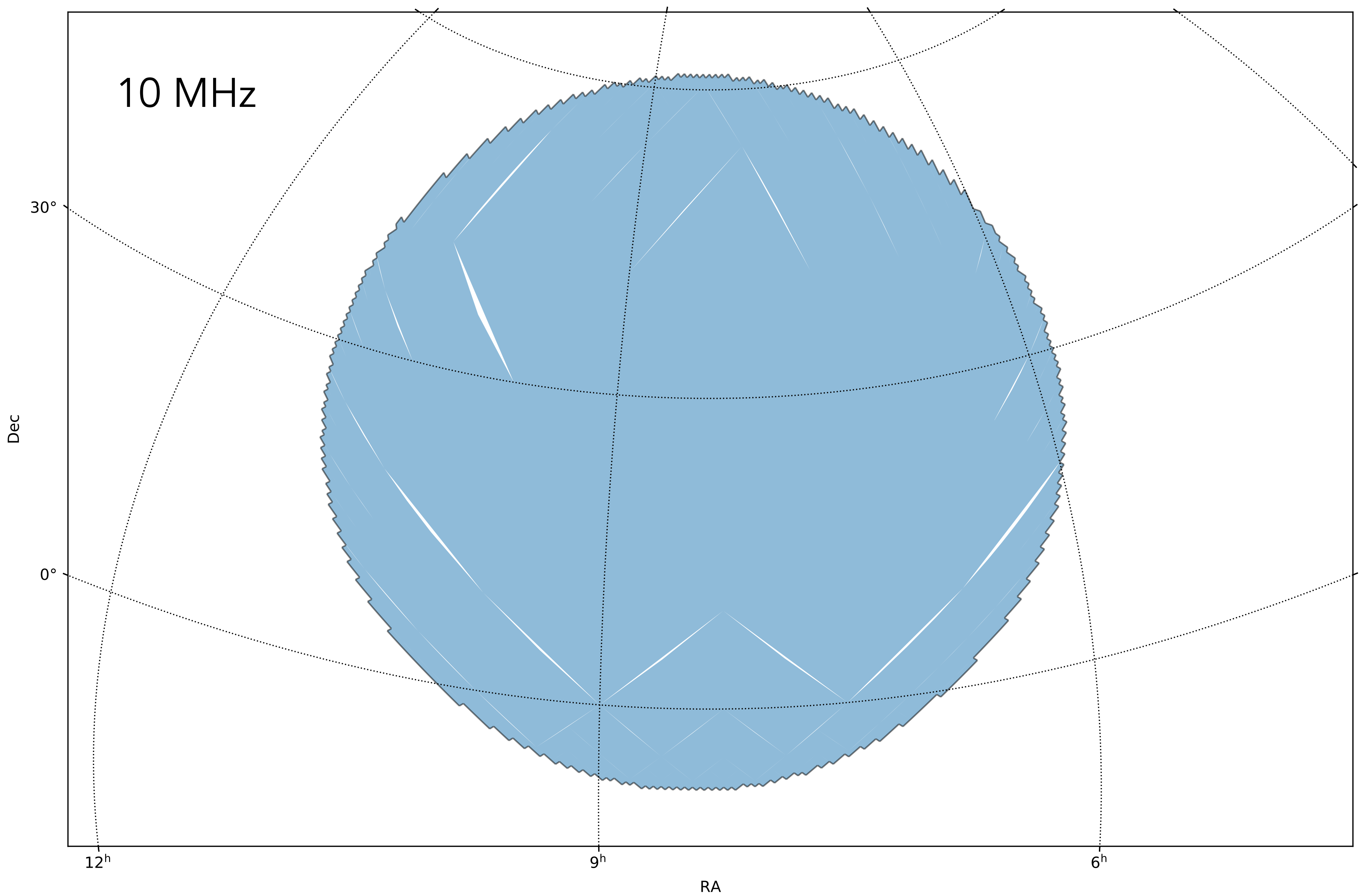
STMOC from NenuFAR database

```
from mocpy import STMOC, MOC
from cdshealpix import cone_search
import astropy.units as u
import numpy as np

def make_stmoc_cones(table, dt, ds, freq):
    """
    """
    mocs = []
    for i in range(len(table)):
        pix, depth, fully_covered_flags = cone_search(
            lon=table['s_ra'][i] * u.deg,
            lat=table['s_dec'][i] * u.deg,
            radius=ma_resol(freq) / 2,
            depth=sresol2order(ds),
            flat=True
        )
        mocs.append(
            MOC.from_healpix_cells(pix, depth, fully_covered_flags)
        )

    stmoc = STMOC()
    stmoc.from_spatial_coverages(
        times_start=Time(table['t_min'], format='mjd'),
        times_end=Time(table['t_max'], format='mjd'),
        spatial_coverages=mocs,
        time_depth=tresol2order(dt),
   )
   return stmoc
```

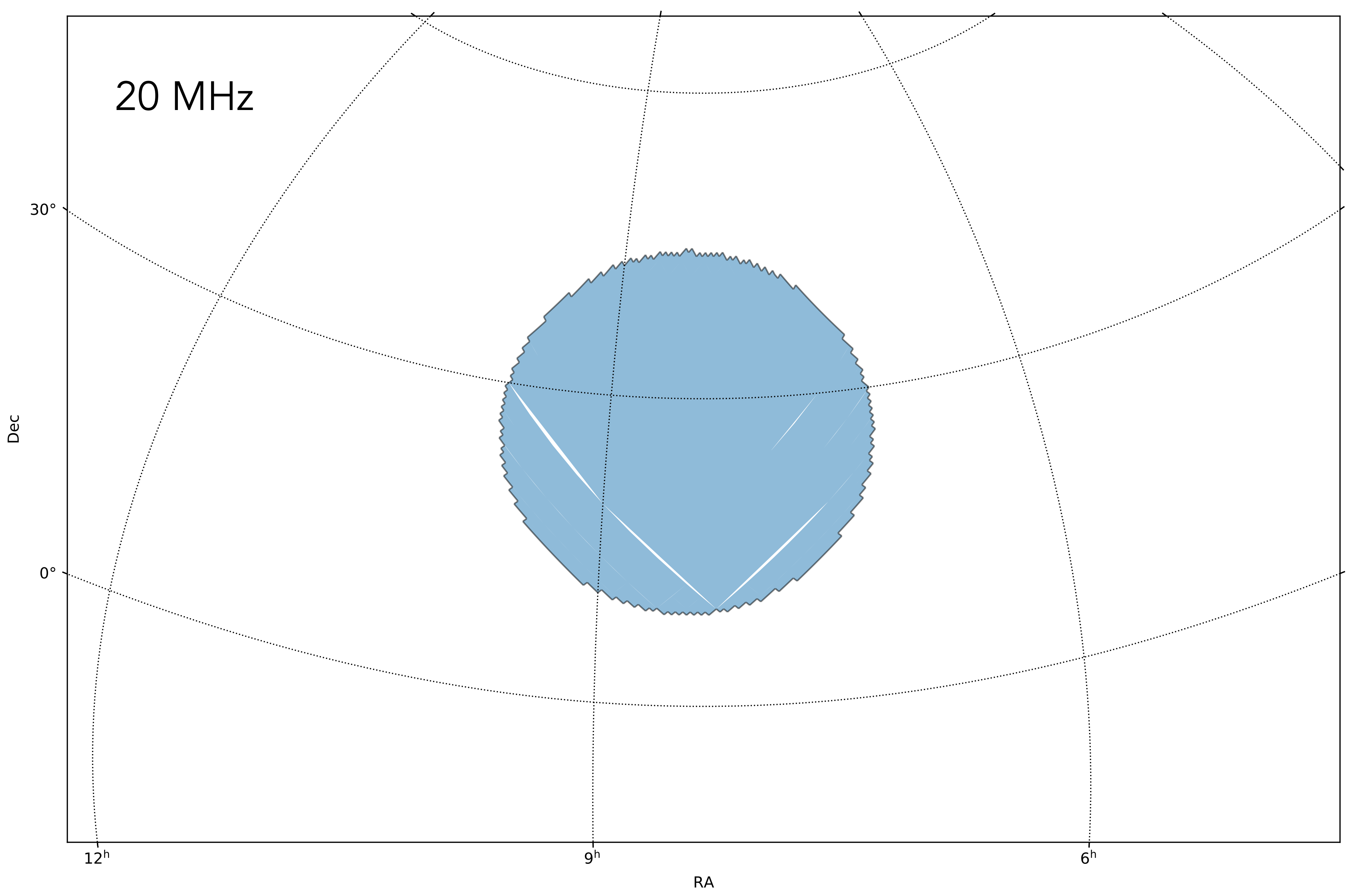
Create a MOC per NenuFAR observation

Build STMOC given a list of MOCS and the time ranges

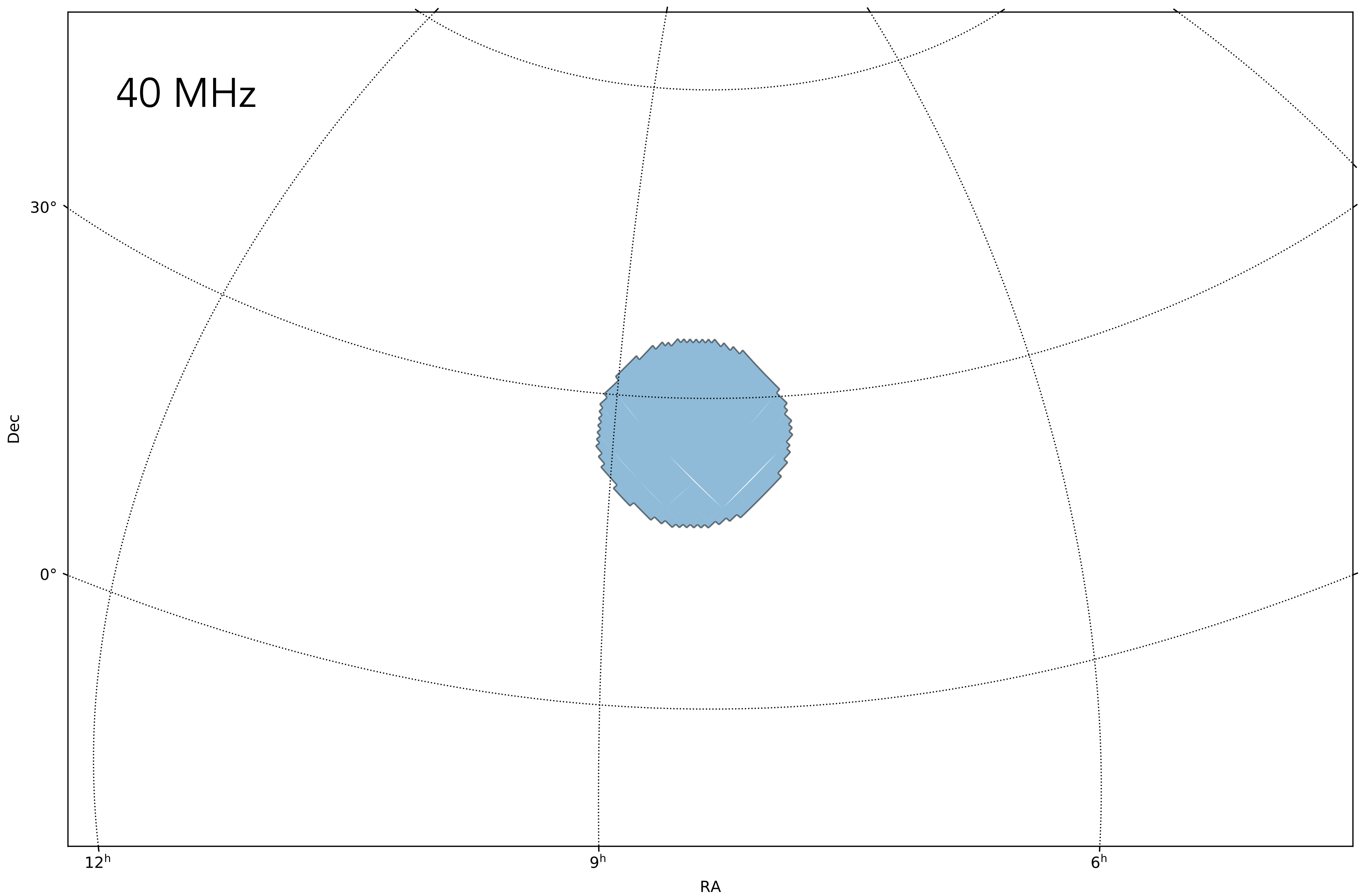
Primary Beam MOC vs. Frequency



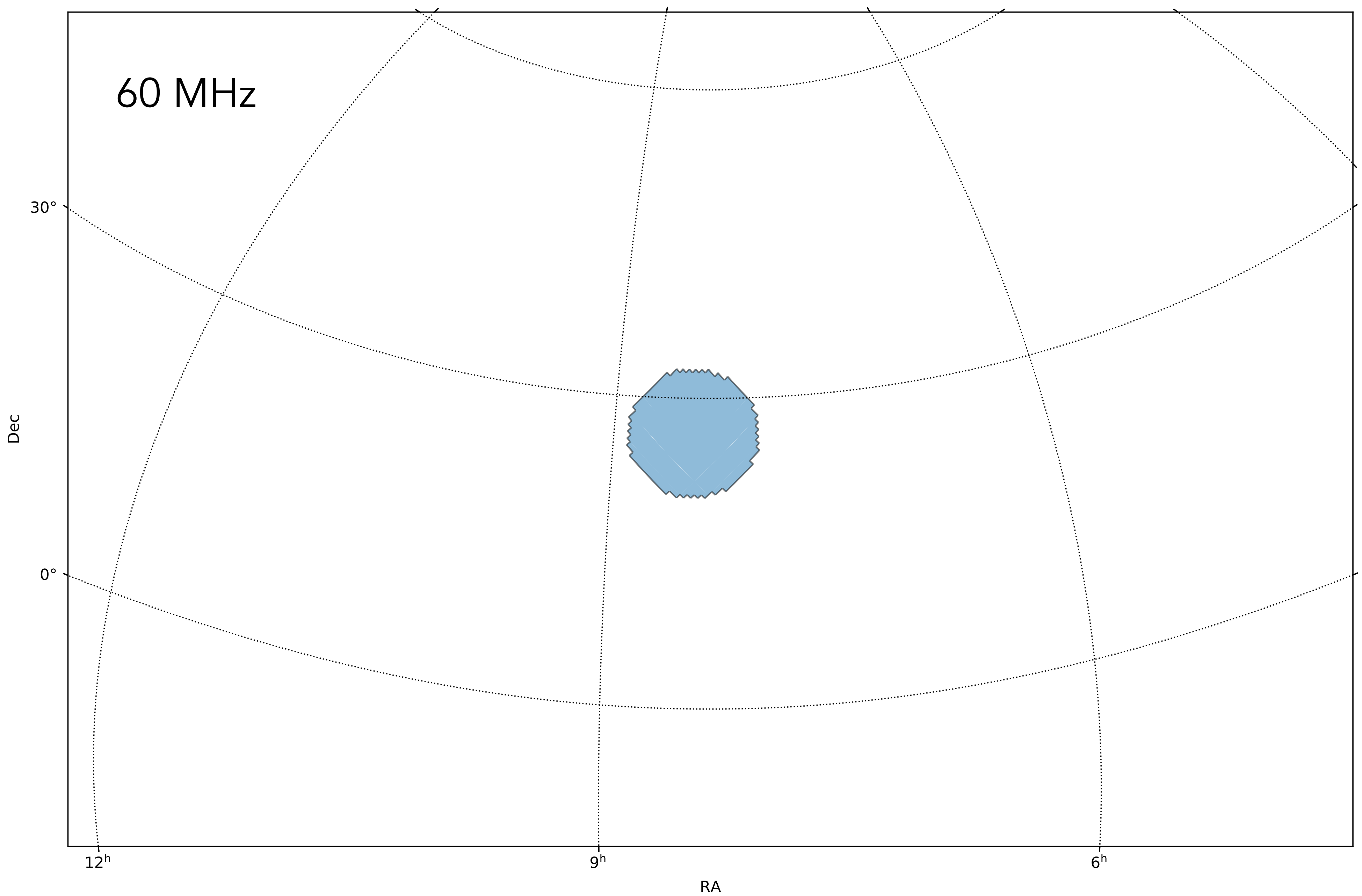
Primary Beam MOC vs. Frequency



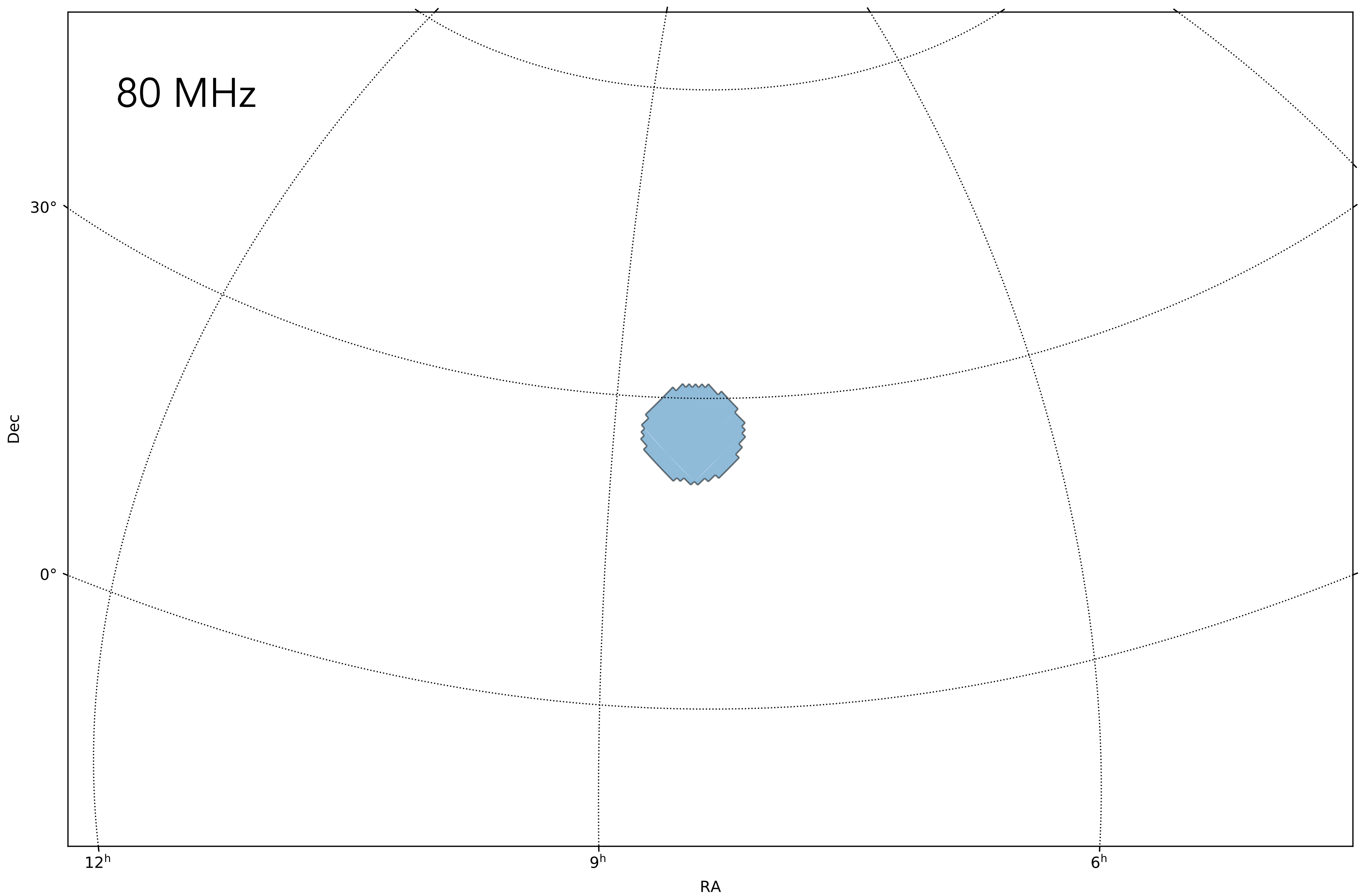
Primary Beam MOC vs. Frequency



Primary Beam MOC vs. Frequency



Primary Beam MOC vs. Frequency



Conclusions

- Easy to build MOCs and TMOCs from NenuFAR observation metadata
- They would be used for:
  - Database exploration (search for transient event, less-polluted datasets)
  - Observation preparation (avoid bright sources in the beam)
- As our frequency range is spread over ~10 - 100 MHz, we have a factor ~10 in primary beam sizes/grating lobe positions --> ST-E-MOC
  - Bright sources inclusion in the beam depends on the frequency!
  - We will soon get the results from a statistical study on RFI contaminated frequency channels
- NenuFAR Python package used in this presentation: *https://github.com/AlanLoh/nenupy*