F-MOC
Towards a frequency MOC ?

IVOA Interop – Bologna – 8-12 May 2023

P. Fernique, F.X.Pineau, B.Cecconi , F.Bonnarel, D.Durand
& all other contributors

For newcomers, a MOC...

Abstract

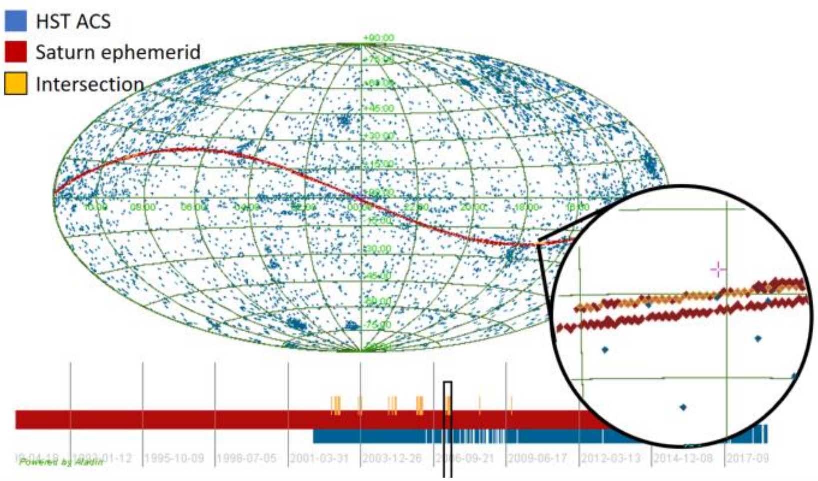
This document describes the Multi-Order Coverage map method (MOC) version 2.0 to specify arbitrary coverages for sky regions and/or time coverages and potentially other dimensions. The goal is to be able to provide a very fast comparison mechanism between coverages. The mechanism is based on a discretization of space and time dimensions. The system is based on the definition of a specific storage of the map coverage using predefined cells hierarchically grouped which makes it easy to produce and use for exploring astronomical collections. There are already a few applications and libraries which are taking advantage of this new standard.

... specifies arbitrary **coverages** for **sky regions** and/or **time coverages**...

... provides a **very fast comparison** mechanism...

... is based on a discretization of space, resp. time, dimensions...

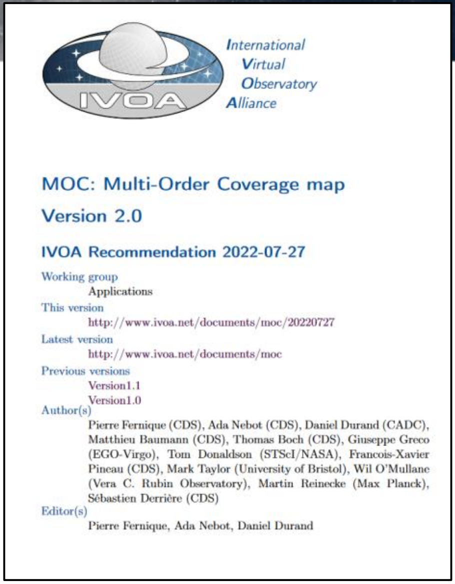
... is based on specific storage of the map coverage using predefined cell hierarchically...



See the IVOA MOC 2.0 document for details

MOC recent evolutions

- **Standards**
  - MOC 1.0 => only spatial MOC
  - MOC 1.1 => + ASCII serialization
  - MOC 2.0 => Spatial + Temporal MOC



International Virtual Observatory Alliance

MOC: Multi-Order Coverage map

Version 2.0

IVOA Recommendation 2022-07-27

Working group
Applications

This version
http://www.ivoa.net/documents/moc/20220727

Latest version
http://www.ivoa.net/documents/moc

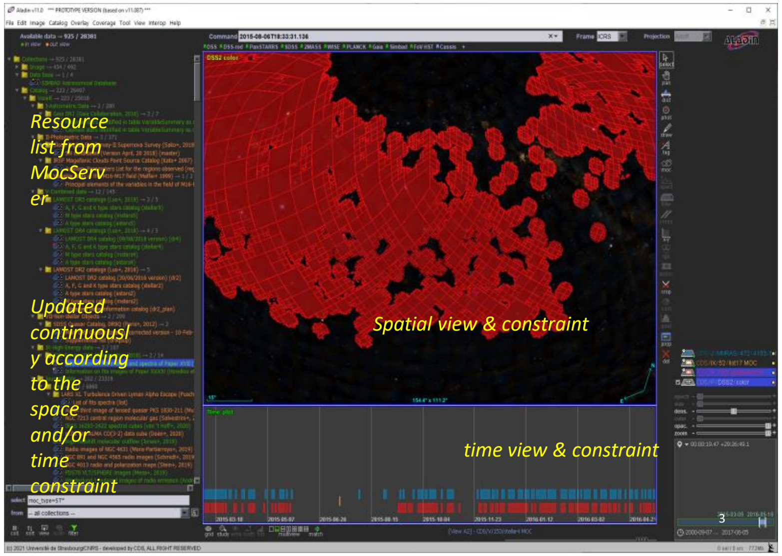
Previous versions
Version1.1
Version1.0

Author(s)
Pierre Fernique (CDS), Ada Nebot (CDS), Daniel Durand (CADC), Matthieu Baumann (CDS), Thomas Boch (CDS), Giuseppe Greco (EGO-Virgo), Tom Donaldson (STScI/NASA), Francois-Xavier Pineau (CDS), Mark Taylor (University of Bristol), Wil O'Mullane (Vera C. Rubin Observatory), Martin Reinecke (Max Planck), Sébastien Derrière (CDS)

Editor(s)
Pierre Fernique, Ada Nebot, Daniel Durand

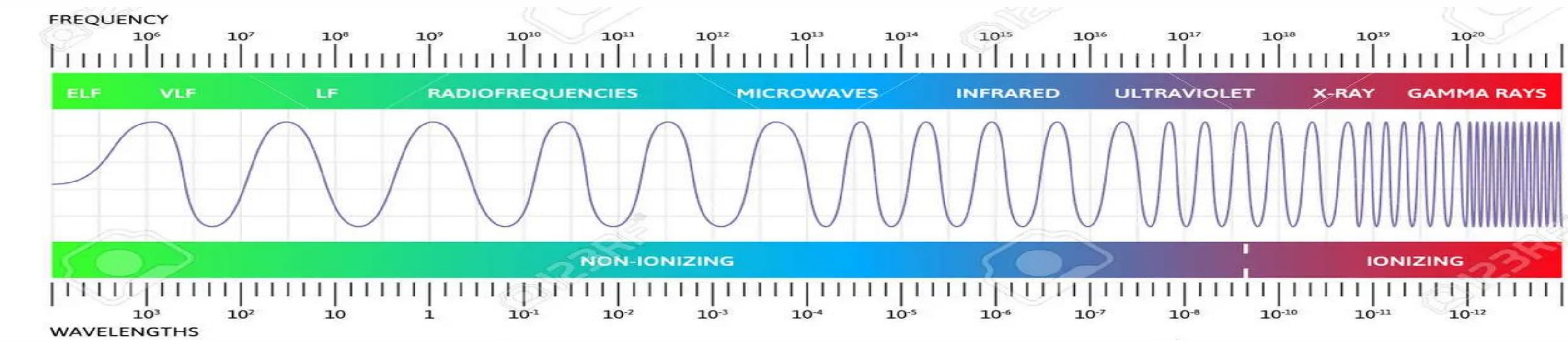
- **Data** from Oct 2021 to March 2023:
  - Spatial MOC: 23,832 -> 26,350
  - Temporal MOC: 1,212 -> 2,575
  - Spatio-temporal MOC: 1,045 -> 1,167

- **Tools** & **libraries**
  - MOCPy, MOC java
  - VO registry, MocServer, …
  - Aladin desktop, ESAsky, …



Space, Time… what about Energy ?

- The **goal** : reuse the same MOC principles to handle **coverages** on the **electromagnetic axis**



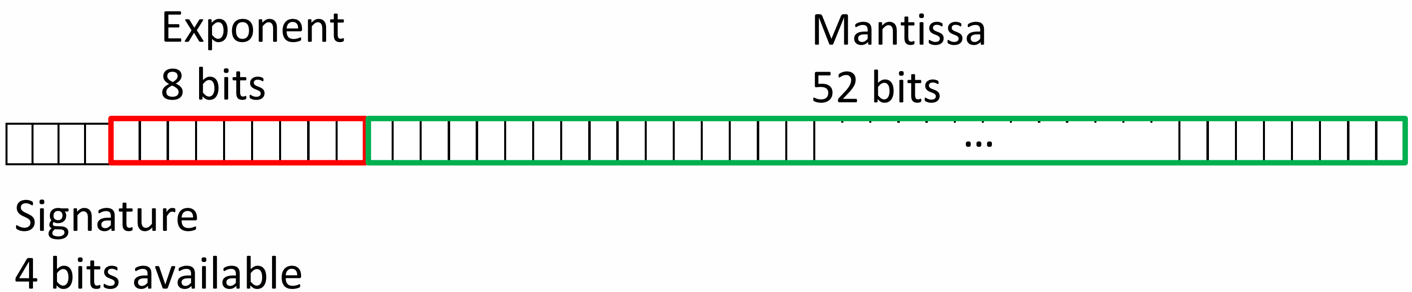
- **Questions** :
  - Energy, wavelength or frequency?
  - How to map these values in a MOC ?

The challenges

- **Reminder**: The MOC only handles **64-bit integer** lists
  - Space: HEALpix indices
  - Time : JD in µs for time
  - Energy : ??
- **Constraints**
  - Amplitude large enough to describe the observations
  - Good accuracy whatever the regime
- **Energy**
  - Difficult to code/represent on a linear axis

The idea (F.X.Pineau & B.Cecconi)

- Use **frequencies**
- Map values as a **logarithmic** expression, using the same principle as the coding of real numbers : mantissa and exponent
  - **52** bits for mantissa
  - **8** bits for exponent (not 11)
  - Save **4** bits for signature



Consequences

- Full observed electromagnetic axis can be covered
- Both internal MOC management are supported:
  - **By ranges** at the deepest order : **[val1..val2[**
  - Or **by hierarchical cells** : **order/val**
- **59** orders
  - As for the time axis, the n-1 order is 2 times less accurate than the n order…
  - … and the corresponding value is divided by 2

The magic F.X. formula

```
long getHash(double freq) {
    long freq_bits = Double.doubleToLongBits(freq);
    long exponent = (freq_bits & F64_EXPONENT_BIT_MASK) >> 52;
    exponent = (exponent - 929) << 52;
    long hash = (freq_bits & F64_BUT_EXPONENT_BIT_MASK) | exponent;
    return hash;
  }
```

```
double getFreq(long hash) {
    long exponent = (hash & F64_EXPONENT_BIT_MASK) >> 52;
    exponent = (exponent + 929) << 52;
    long freqBits = (hash & F64_BUT_EXPONENT_BIT_MASK) | exponent;
    double freq = Double.longBitsToDouble(freqBits);
    return freq;
  }
```

The results

- **Very Fast** mapping
- Resulting **amplitude** (in Hz)
  - FREQ_MIN = **5.048709793414476e-29**
  - FREQ_MAX = **5.846006549323611e+48**
- **Accuracy**
  - Variable

  => depending on the frequency value

What does this mean for the various regimes?

Regime	Avg.freq	Res.max (59)	freq/dFreq
em.radio.100-200MHz	150MHz/1.995m	29.802nHz/4.1E-13m	5.03E15
em.radio.6-12GHz	10GHz/29.93mm	1.907オHz/6.5E-15m	5.24E15
em.mm.20-100GHz	60GHz/4.988mm	7.629オHz/8.1E-16m	7.86E15
em.mm.750-1500GHz	1.125THz/266.042um	244.141オHz/5.0E-17m	4.608E15
em.IR.30-60um	7.5THz/39.906um	976.562オHz/6.3E-18m	7.68E15
em.IR.3-4um	87.5THz/3.421um	15.625mHz/7.9E-19m	5.6E15
em.opt.R	450THz/665.105nm	62.5mHz/9.9E-20m	7.2E15
em.opt.B	675THz/443.404nm	125mHz/4.9E-20m	5.41E15
em.UV.100-200nm	2.25PHz/133.021nm	250mHz/2.5E-20m	9E15
em.UV.10-50nm	18PHz/16.628nm	2Hz/3.1E-21m	9E15
em.X-ray.soft	265PHz/1.129nm	32Hz/1.9E-22m	8.28E15
em.X-ray.hard	16.5EHz/18.139pm	2.048kHz/3.0E-24m	8.07E15
em.gamma.soft	615EHz/486.663fm	131.072kHz/1.9E-25m	4.7E15
em.gamma.hard	2,000EHz/149.649fm	262.144kHz/2.4E-26m	7.63E15

Prototype implementations

- **MOC java** (P. Fernique) : done
  - F-MOC: operations + serializations
  - 2D extensions ready:
    - Space Frequency MOC (SFMOC)
    - Frequency Time MOC (FTMOC)
- **MOCpy** (F.X. Pineau): in progress
  - Already available in MOC-cli (RUST lib used by MOCpy)
  - F-MOC: operations + serializations

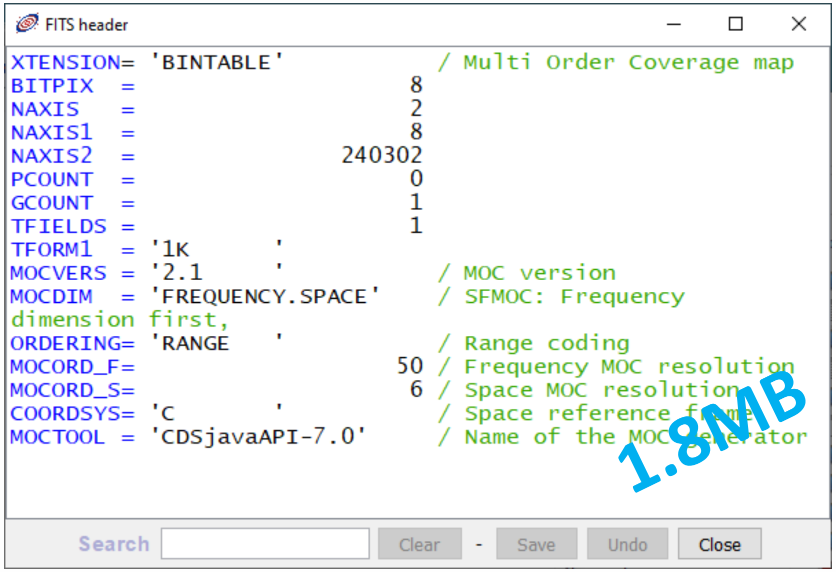
Technical tests

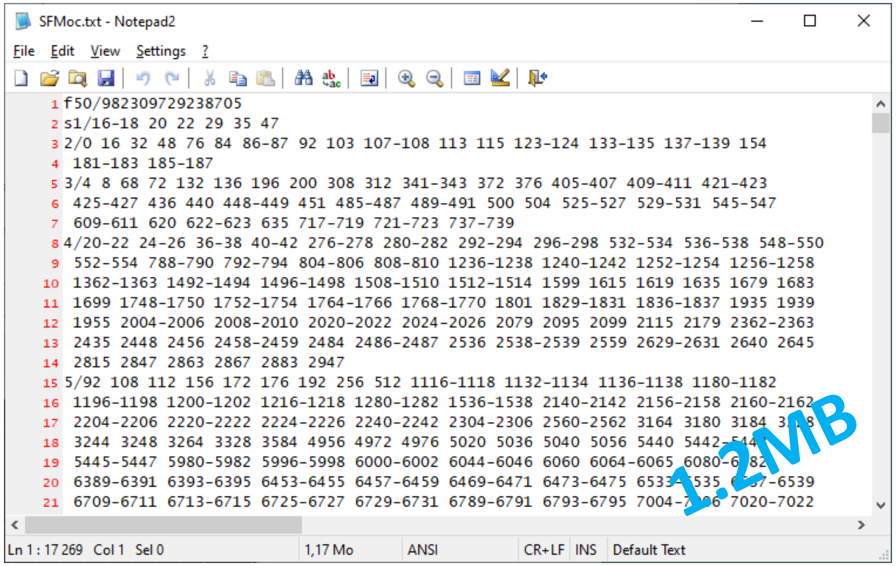
- The first **F-MOC** and **SF-MOC**

  => Build from the 973 HiPS based on **SMOC** and **em_min, em_max** interval (F-order=50, S-order=6)

  https://aladin.cds.unistra.fr/moc/FMOC.fits or FMOC.txt

  https://aladin.cds.unistra.fr/moc/SFMOC.fits or SFMOC.txt





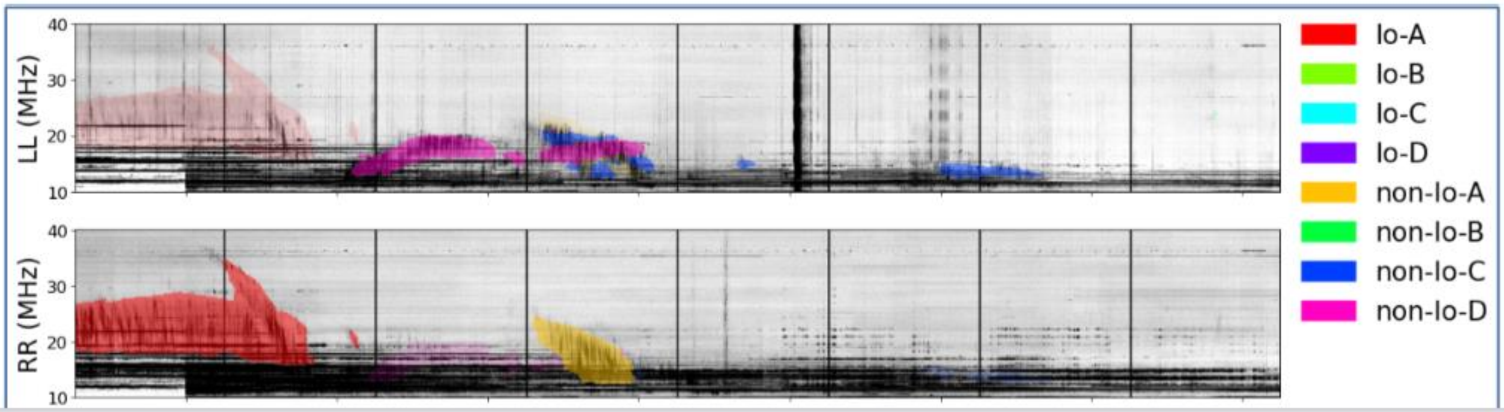
Scientific use cases

=> D.Durand

FMOC & SFMOC associated to HST HiPS

=> B.Cecconi

Lots of ideas to be tested…



Jupiter notebook playing with FMOC (B.Cecconi *+ M.Marchand*)

508 lines (508 sloc) | 23.1 KB

Raw Blame

First steps with Frequency MOCs

In [1]:
```
# Standard Library
from pathlib import Path

# General and astronomy packages
import numpy as np
from astropy.units import Unit
from maser.data import Data

# Specific to FMOCs
from mocpy.fmoc import FrequencyMOC
```

We use a file from the Cassini/RPWS/HFR database. This radio instrument has a configurable spectral sampling. The data file is a level 2 data file, containing the centers and widths of each spectral bin.

The file (and many others) is available for download here: https://lesia.obspm.fr/kronos/data/2012_091_180/n2/

In [2]:
```
file = Path("../resources/FMOC/P2012180.20")
```

We load the data using the `maser.data` module, which recognizes the file

In [3]:
```
n2 = Data(file)
n2.fields
```

Out[3]: `dict_keys(['ydh', 'num', 't97', 'f', 'dt', 'df', 'autoX', 'autoZ', 'crossR', 'crossI', 'ant'])`

In [4]:
```
n2.dataset
```

Out[4]: `'co_rpws_hfr_kronos_n2'`

Spectral sweeps are available as a generator using the `.sweeps` property.

In [5]:
```
sweep = next(n2.sweeps)
print(f"This sweep has {len(sweep.data)} spectral steps")
sweep.data.dtype
```

Next steps

- Continuing exploratory work
  - Complete the **MOCpy** implementation
  - Implement **scientific** use cases
  - Extend **clients** for using these libraries
    (Aladin Desktop ? CASSIS ?)
- Towards a IVOA MOC 3.0 standard?
  - Extending MOC 2.0 to frequencies should be easy (document already oriented for this)
  - Maybe a bit too early to decide? Or?
  - Volunteers for this new edition step?

- At this stage, **no extension to a 3D MOC** (=SFT-MOC)
  => Too big MOC? New algorithms.