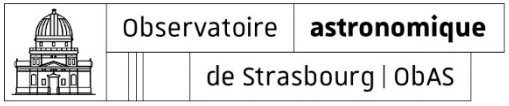
HiPS3D

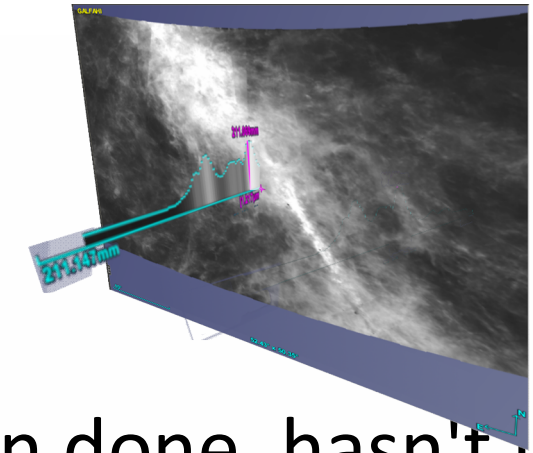
Proposal to extend the IVOA HiPS standard to cubic data

Interop IVOA – College Park – June 2025

Pierre Fernique
M. Allen, M. Baumann, T.Boch, F. Bonnarel,
C. Bot, FX Pineau, K. Voggel



The plan



1. **Context**
2. **HiPS cubic**. It's already been done, hasn't it? What's the problem?
3. **HiPS3D**. A proposed solution
4. **HiPS3D** prototype **demo**
5. **Implications for the IVOA HiPS standard**

Context of this work

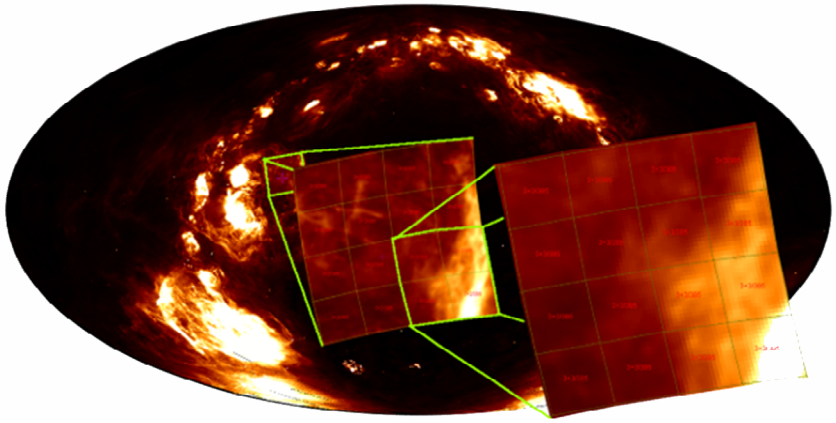
Soon an **avalanche of cubic data** (SKA =300PB of data/year in the form of cubes of several hundred GB, or even much more).
IVOA has to invente solutions to **extend our tools & standards** to this evolution.

3 years of discussions, tests, studies and developments supported by the **CDS** in its contribution to the **Orange SKA** (visualization) & **SKA SRCnet FR** team.

HiPS – What is it?

- The **Hierarchical Progressive Survey** method
- Standardized by **IVOA in 2017**
- Based on **HEALpix resampling**
- Makes **a sky survey** accessible, visualizable and even manipulable, **whatever the size** of the survey, **the quality** of the network and the **computing power** available to the astronomer.

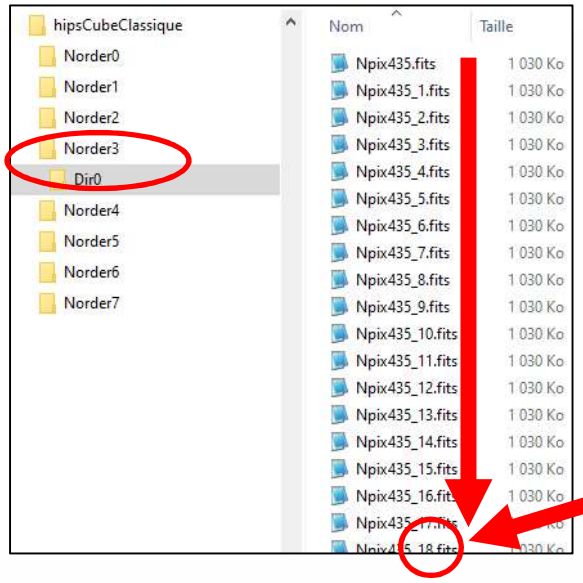
- A response to the big data challenge
- **Implemented** by several scientific, amateur and public visualization tools: Aladin, hips2fits, ESAsky, ESO portal, WWT, Firefly, DIGISTAR, RSACosmos, Stellarium...



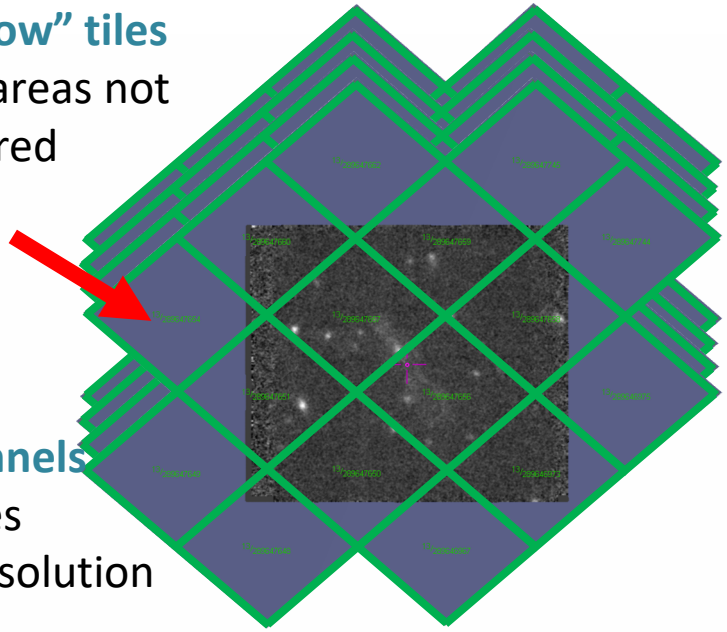
HiPS cube: what's the problem?

- IVOA 2017 standard: a **HiPS cube is built as a collection of HiPS images**
  = 1 HiPS per channel.
- Consequences:
  1. The **original cubes** must be **homogeneous**, i.e. have the same number of channels, and each channel must represent the same physical quantity;
  2. The **number of files**, **volume** and **processing time** of the final HiPS are linearly proportional **to the number of channels**;
  3. The case of "**hollow HiPS tiles**" is becoming very costly (too much wasted space)..

In concrete terms...



"Hollow" tiles
blue areas not required



Number of channels
= Number of tiles whatever the resolution

Example	Number of files/tiles	Volume in GB	Processing time
1 Cube 351x301x**3681**	1	1,45	-
Classic HiPS cube	66261	65,04	6mn 26s
HiPS 3D	**4634 (/15)**	**2,28 (/30)**	**36s (/12)**

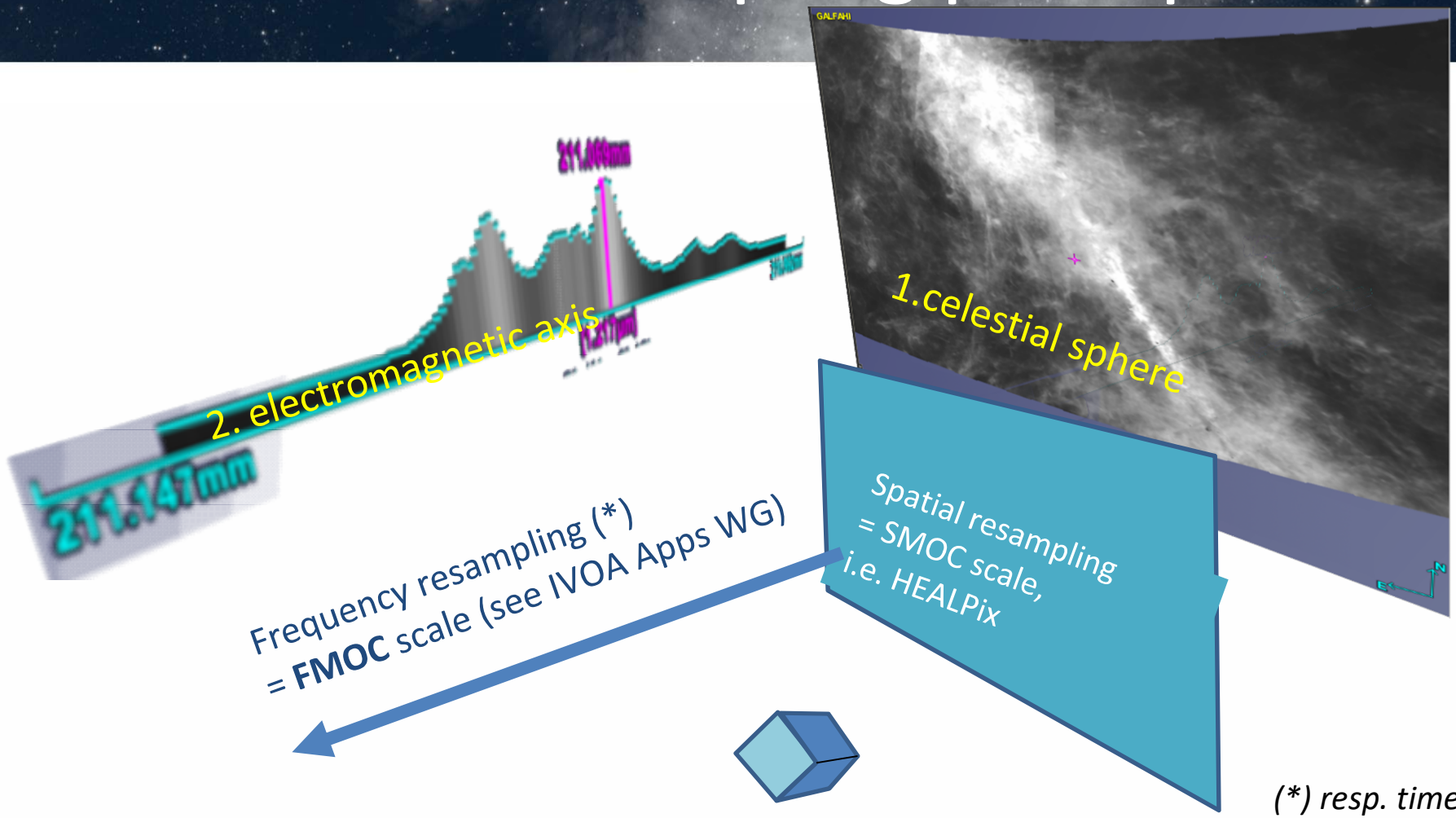
Proposed solution: HiPS3D(*)

*" Extend the idea of the current HiPS cube towards a "real" **HiPS 3D**, i.e. **hierarchical** in each physical dimension and based on **absolute physical units and reference frames**. "*

In other words: **discretize** not just the surface of the **sphere**, but also a **thickness**, and do this at several resolutions.

(*) HiPS3D already briefly presented at Interop + ADASS Malta 2024

HiPS3D resampling principle



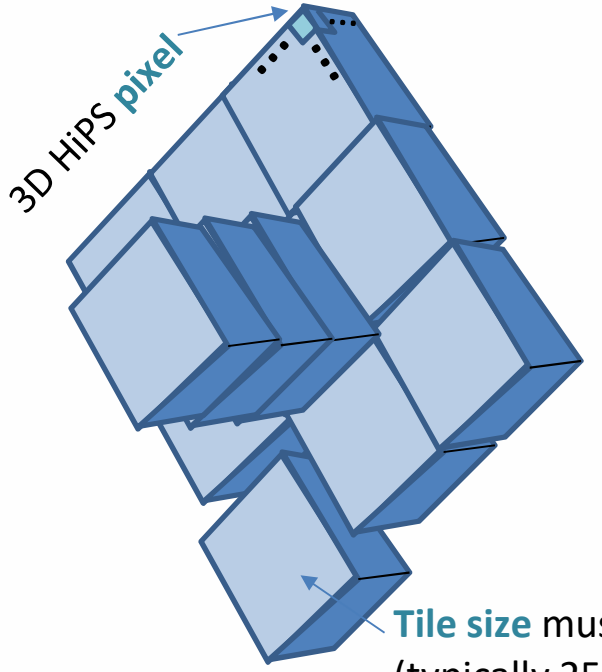
Each 3D pixel has a unique HiPS3D address
= (OrderS/NpixS, OrderF/NpixF)

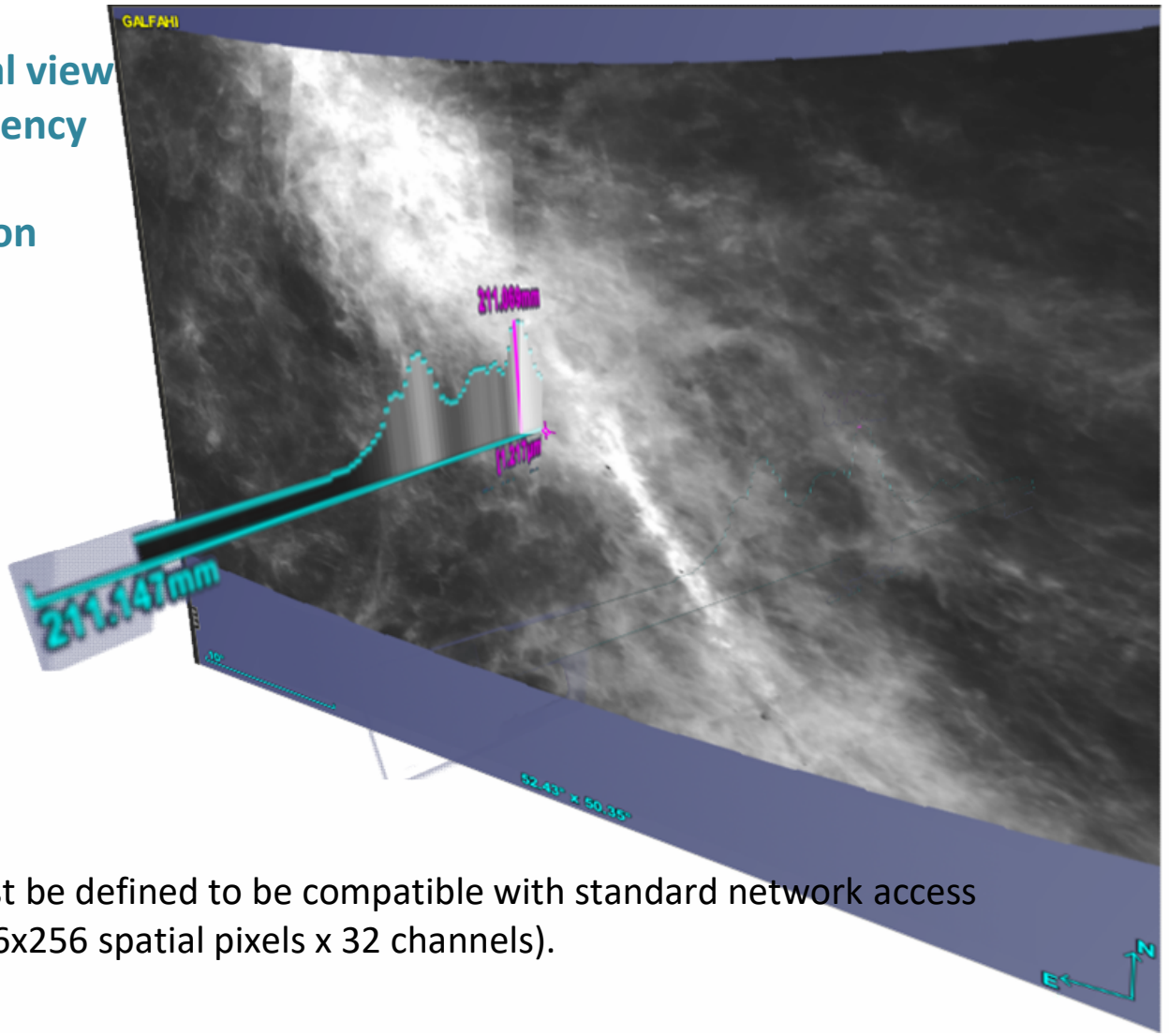
*(*) resp. time resampling based on **TMOC** scale*

HiPS3D “ tiling ” principle

The HiPS3D client loads :

- the tiles covering the **spatial view**
- The tiles covering the **frequency view**
- at the **appropriate resolution**





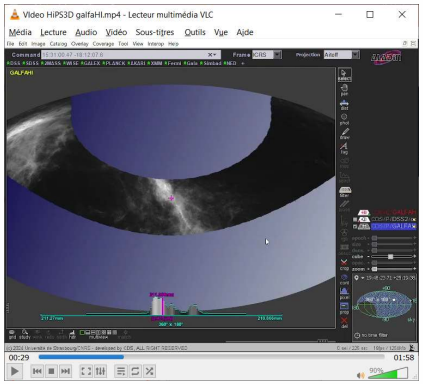
Tile size must be defined to be compatible with standard network access (typically 256x256 spatial pixels x 32 channels).

Implementation tests

Proof-of-concept developments starting in June 2024
Space-frequency data oriented (not yet on spatio-temporal data)

1. Extended "**Hipsgen**" HiPS generator *(–hips3d param)*
   - Takes into account FITS cubes in **frequencies**, **wavelengths** or **speeds;**
   - Successfully tested on cube sets :
     - **MUSE**: 2600 cubes (350x350x3700 = 4TB) – pointed obs.
     - **SKADC2**: 1 cube (5851x5851x6668 = 850GB) - simulation
     - **ASKAP**: 4 cubes (11000x11000x144 = 177GB) - pointed obs.
     - **GALFAHI**: 225 cubes (512x512x2048 = 225GB) – spatial mosaic
     - **MEERKAT**: 3 cubes (5000x5000x2000 = 310GB) – "Frequency "mosaic
     - **ALMA**: 560 cubes (heterogeneous = 1.4TB) – pointed obs. in "space & frequency" mosaic
2. **Aladin Desktop** HiPS client extended to HiPS3D *(version proto 12.6)*
3. HiPS server-side tool for dynamic extraction of a high-resolution spectrum from a HiPS3D (function similar to Hips2fits)

HiPS3D demo/tutorial



HiPS3D frequency discovery tutorial

Centre de Données astronomiques de Strasbourg
Auteur : Pierre Fernique
V1.98 – 26 mai 2025

- Version française : https://aladin.cds.unistra.fr/java/TutoHiPS3D.pdf
- English version: https://aladin.cds.unistra.fr/java/TutoHiPS3Den.pdf

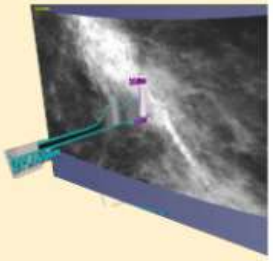
The aim of this tutorial is to introduce you to the possibilities offered by the new HiPS3Ds implemented by CDS over the last few weeks, which can be manipulated with the latest prototype version of Aladin Desktop.

Please note that this is an R&D version, and therefore not a final prototype (still bugs, functions that don't work yet, or not like before). So please do not use this version for anything other than this tutorial (and certainly not distribute it without informing the recipient).

First of all, what is a HiPS3D?

A HiPS3D is a generalization of HiPS that allows you to walk around in a "cubic" mosaic of observations. Instruments like MUSE, ASKAP or SKA produce data cubes, not images. HiPS3D takes this third dimension into account, allowing you to pan and zoom both spatially (as with conventional HiPS) and in frequency (a new feature).

Note that extension to "temporal" cubes is planned (Rubin observations, for example).



If you don't have the time or the inclination to do this tutorial, you can just watch this video => https://aladin.cds.unistra.fr/java/HiPS3D-apr25.mp4

Once you've finished this tutorial, please don't hesitate to send us feedback (cds-question@astro.unistra.fr) with your suggestions, reviews and encouragement, as this will be very useful to us. Thanks for your time.

Here we go with the tutorial, which should take you no more than 10 minutes... but more if you enjoy it!

Requirements

All you need is the "good" proto version of Aladin Desktop (at least **v12.620**).
=> https://aladin.cds.unistra.fr/java/AladinProto.jar

https://aladin.cds.unistra.fr/java/TutoHiPS3Den.pdf

Proposal for HiPS standard evolution:
Tile access API

- **Keep the same idea as for the existing HiPS**, adding only the "_x" suffixes associated with the additional dimension.
- Still possible to use a classic **File System** for storage, and an **HTTP server** for distribution.

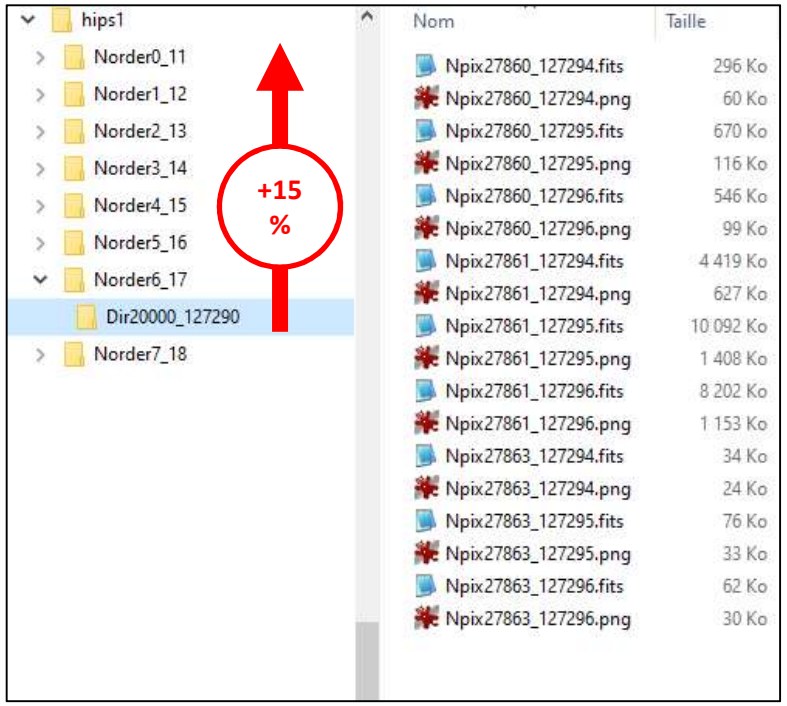
http[s]://server/path/…/Norder***K_L***/Dir***D_E***/Npix***N_M***.*ext*

- **K** is the spatial order, **L** frequency order,
- **N** spatial index *(SMOC method= HEALPix),*
  M frequency index *(FMOC method – see appendix).*
- **D** = (N/10000)*10000, **E**= (M/10)*10 *(integer division)*

Example: https://alasky.cds.unistra.fr/GALFAHI/GALFAHI-Narrow-DR2-3D/
Norder**3**_**21**/Dir**0**_**2036900**/Npix**218**_**2036905**.fits

HiPS3D directory structure

- Principle: **allow all combinations of resolution** (spatial vs. frequency)(*)
- But the intuition that a single combination will be enough = **simultaneous reduction in spatial and frequency resolution**(*)

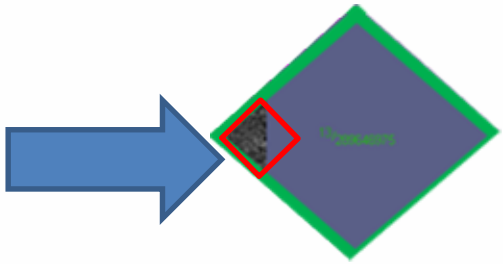


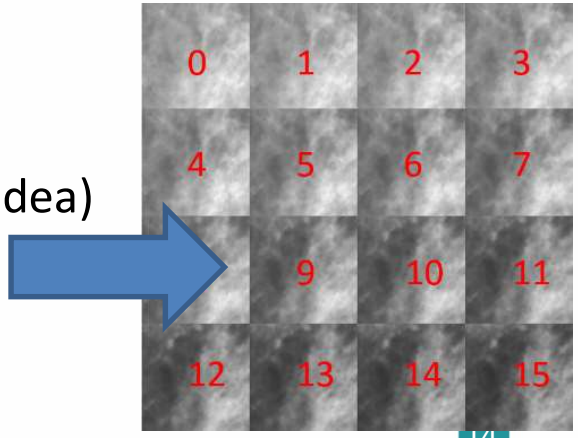
As a result, the hierarchy **adds only 15%** to the volume of HiPS (reduced by a factor of 8 for each sub-order)..

(*) resp. spatial vs time

HiPS3D tile formats

- Still **classic files**
- But **cubic tiles**
- Keep up the idea of **several possible tile sets**, depending on use:
  - Full dynamic => cube **FITS tiles**
    TRIM method proposal *(remove blank margins)*
    to reduce the “hollow tile” effect,
    even lossless compression (e.g. RICE)
  - Quick view => **Compressed tiles**
    proposal for use of conventional compressors
    (jpeg, png, webp…) in **checkerboard mode** (T.Boch idea)
    Note: disappointing video compressors
    (slower (x5-10), not always alpha channel)



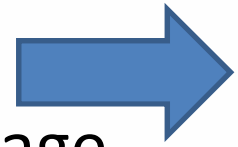


Metadata evolution

- Addition of the 3 keywords required for the new dimension in the HiPS "**properties**" file:
  Exemple en fréquence:
  - **hips_tile_depth**
  - **hips_order_freq**
  - **dataproduct_type** = spectral-cube

```
creator_did          = ivo://CDS/P/Muse
obs_title            = Muse
hips_builder         = Aladin/HipsGen v12.510
hips_version         = 1.5
hips_frame           = equatorial
hips_order           = 14
hips_order_freq      = 16
hips_order_min       = 0
hips_tile_width      = 256
hips_tile_depth      = 16
hips_tile_format     = png fits
dataproduct_type     = spectral-cube
em_min               = 4.741906994477967E-7
em_max               = 9.353675889622063E-7
moc_sky_fraction     = 6.208E-10
…
```

- Usage of **SFMOC**(*) to describe space-frequency coverage



Standardize FMOCs and SFMOCs
(cf Interop IVOA 2024 presentation)

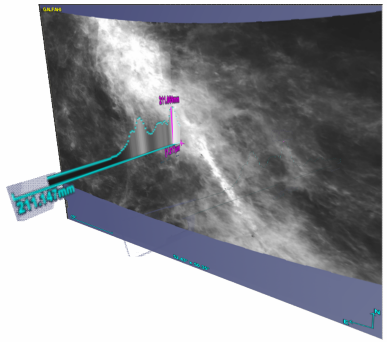
- At a minimum, impose keywords **em_min, em_max**(*)

(*) resp. STMOC, t_min, t_max

What's next?

More tests, more **testers** => please try the tuto
Support in **Aladin Lite** => in progress

Start a **new IVOA HiPS version** => **2.0** ?
(in the end, not so many changes)



Thanks to everyone already involved!
Thanks to the SKA project
for all the brainstorming!

Appendix 1: Vocabulary (proposal)

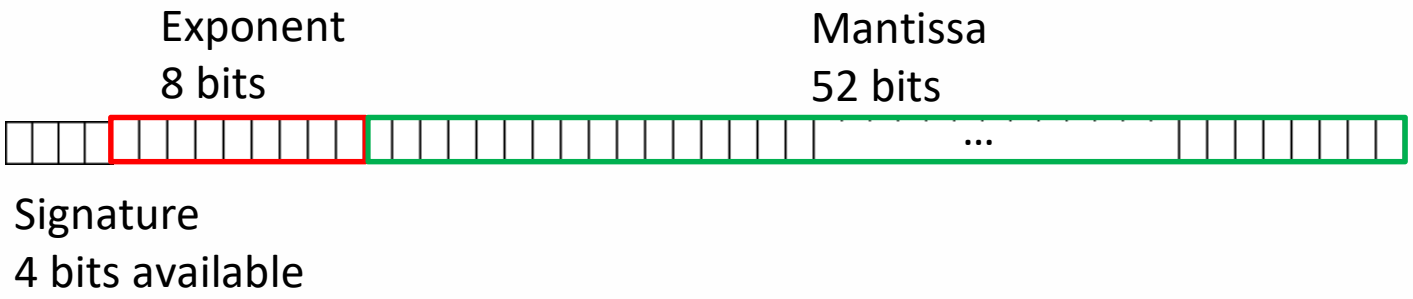
- **HiPS**: Hierarchical partitioning of the surface of a sphere (based on SMOC discretization = HEALPix)
- **HiPS3D**: Hierarchical partitioning of a sphere whose surface has thickness
  - **HiPS3D-time**: thickness is partitioned using linear discretization (TMOC scale)
  - **HiPS3D-freq**: idem but with a logarithmic method (FMOC scale)
- **pixel**: minimal HiPS partitioning element
  - Spatial case: HEALPix diamond (constant surface for a given order)
  - Temporal case: time interval (constant duration for a given order)
  - Frequency case: Frequency interval (logarithmic interval for a given order)
- **tile**: group of "adjacent" HiPS pixels (according to HiPS rule)
  - **width**: number of linearly grouped pixels (in time and frequency), and number of pixels on one side (in space)
  - **widthZ**: specifically for the frequency or time dimension in HiPS3D
  - **x[,y]**: Cartesian coordinate of a pixel in a tile
  - **channel**: channel number for thick tile
- **address**: Composite number used to identify a HiPS pixel or tile. It is composed of 2 values: order and npix.
  - **order**: HiPS order
  - **npix**: the pixel index in the specified order
    In the case of HiPS3D, 2 additional values will be added to the address:
  - **orderZ**: frequency (respectively time) order
  - **npixZ**: the pixel's frequency (or time) index

    The ASCII syntax of the address is as follows: **order/npix** and **order/npix_orderZ/npixZ** for HiPS3D.

Appendix 2: Frequency discretization
(Idea: FX.Pineau & B.Cecconi)

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



Already presented at Interop 2024 for MOC frequency extension

The magic FMOC 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;
  }
```