HiPS3D

Proposition d'extension du standard IVOA HiPS aux données cubiques

ASOV – Paris – Mars 2025

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

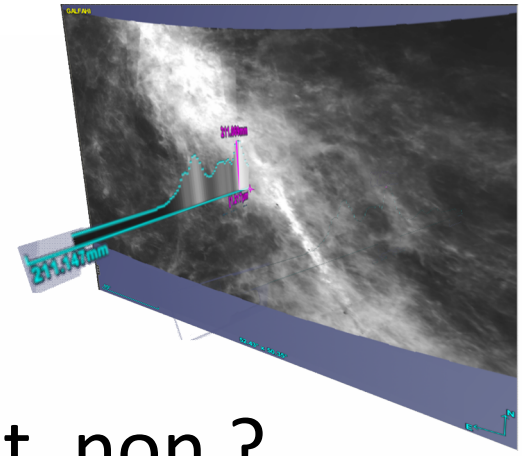
CENTRE DE DONNÉES ASTRONOMIQUES DE STRASBOURG

Observatoire astronomique de Strasbourg | ObAS

cnrs

Université de Strasbourg

Le plan

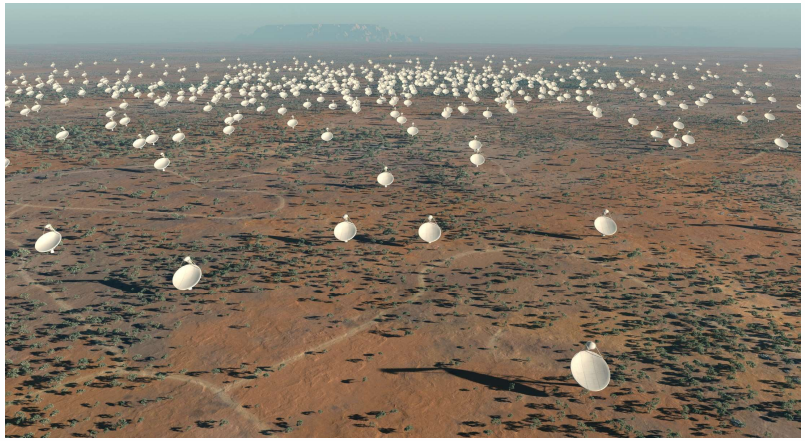


1. **Contexte**
2. **HiPS cubique**. C’est déjà fait, non ?
   Alors ! Quel est le problème ?
3. **HiPS3D**. Une proposition de solution
4. **Démo** d’un prototype HiPS 3D
5. **Implication sur le standard** IVOA HiPS

Contexte de ce travail

Bientôt une **avalanche de données cubiques**
(SKA =300Po de données/an sous forme de cubes de plusieurs centaines de Go, voire bien au-delà)
Inventer les solutions pour **étendre nos outils & standards** à cette évolution.

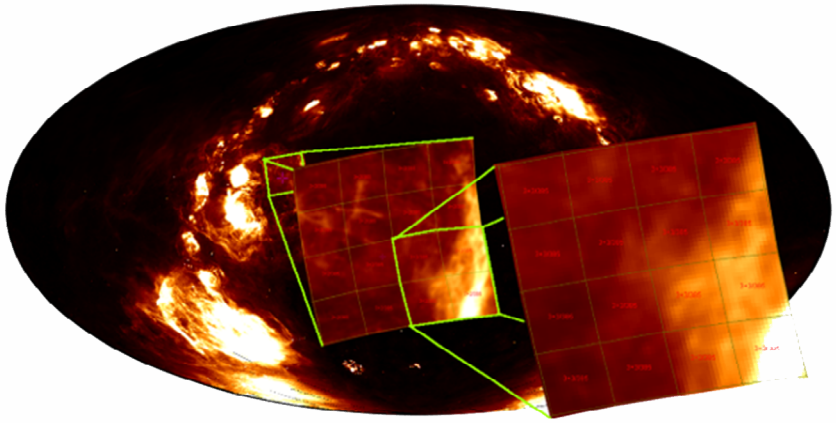
3 ans de discussions, tests, études, développements portés par le **CDS** dans sa **contribution à l'équipe Orange SKA** (visualisation) & **SKA SRCnet FR**



HiPS – C'est quoi de nouveau ?

- La méthode **Hierarchical Progressive Survey**
- Standardisée par l'**IVOA en 2019**
- Basée sur un rééchantillonnage HEALpix
- Rend un **relevé du ciel** accessible, visualisable et même manipulable, **quelle que soit la taille** du relevé, **la qualité** du réseau et **la puissance** de calcul dont dispose l'astronome..

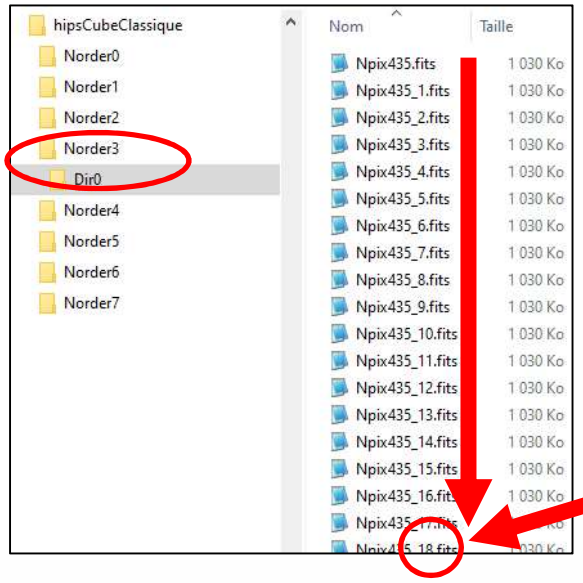
- Une réponse au défi du big data
- **Implémentée** par plusieurs outils de visualisation scientifiques, amateurs et grands publics : Aladin, hips2fits, ESAsky, ESO portal, WWT, Firefly, DIGISTAR, RSACosmos, Stellarium...



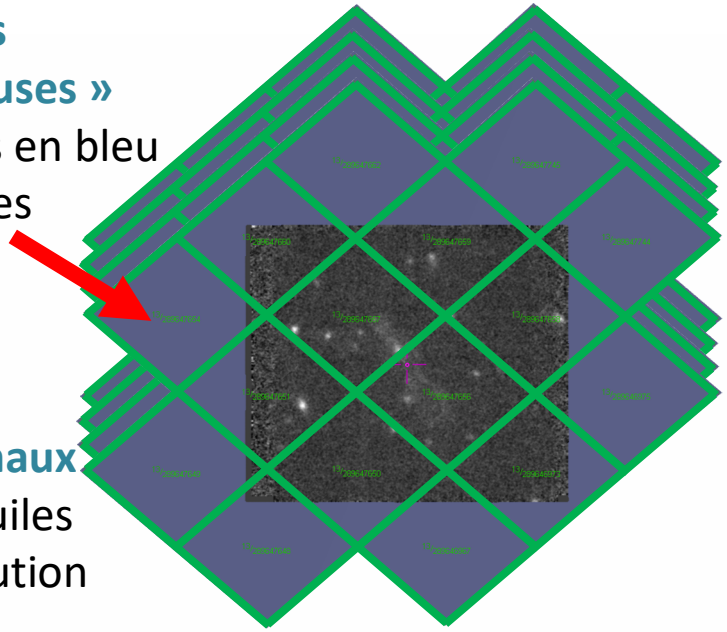
HiPS cube : quel est le problème ?

- Standard IVOA 2017: un **HiPS cube est construit comme une collection de HiPS images**
  = 1 HiPS par canal.
- Conséquences:
  1. Les **cubes originaux** doivent être **homogènes**, càd avoir le même nombre de canaux, et chaque canal doit représenter la même grandeur physique ;
  2. Le **nombre de fichiers**, le **volume** et le **temps** de traitement du HiPS final sont **proportionnels** linéairement **au nombre de canaux** ;
  3. Le **phénomène des "tuiles HiPS creuses"** devient très couteux (trop de place perdue).

Très concrètement…



Tuiles « creuses »
zones en bleu inutiles



Nombre de canaux
= Nombre de tuiles qq soit la résolution

Exemple	Nb de fichiers/tuiles	Volume en Go	Temps de traitement
1 Cube 351x301x**3681**	1	1,45	-
HiPS cube classique	66261	65,04	6mn 26s
HiPS 3D	**4634 (/15)**	**2,28 (/30)**	**36s (/360)**

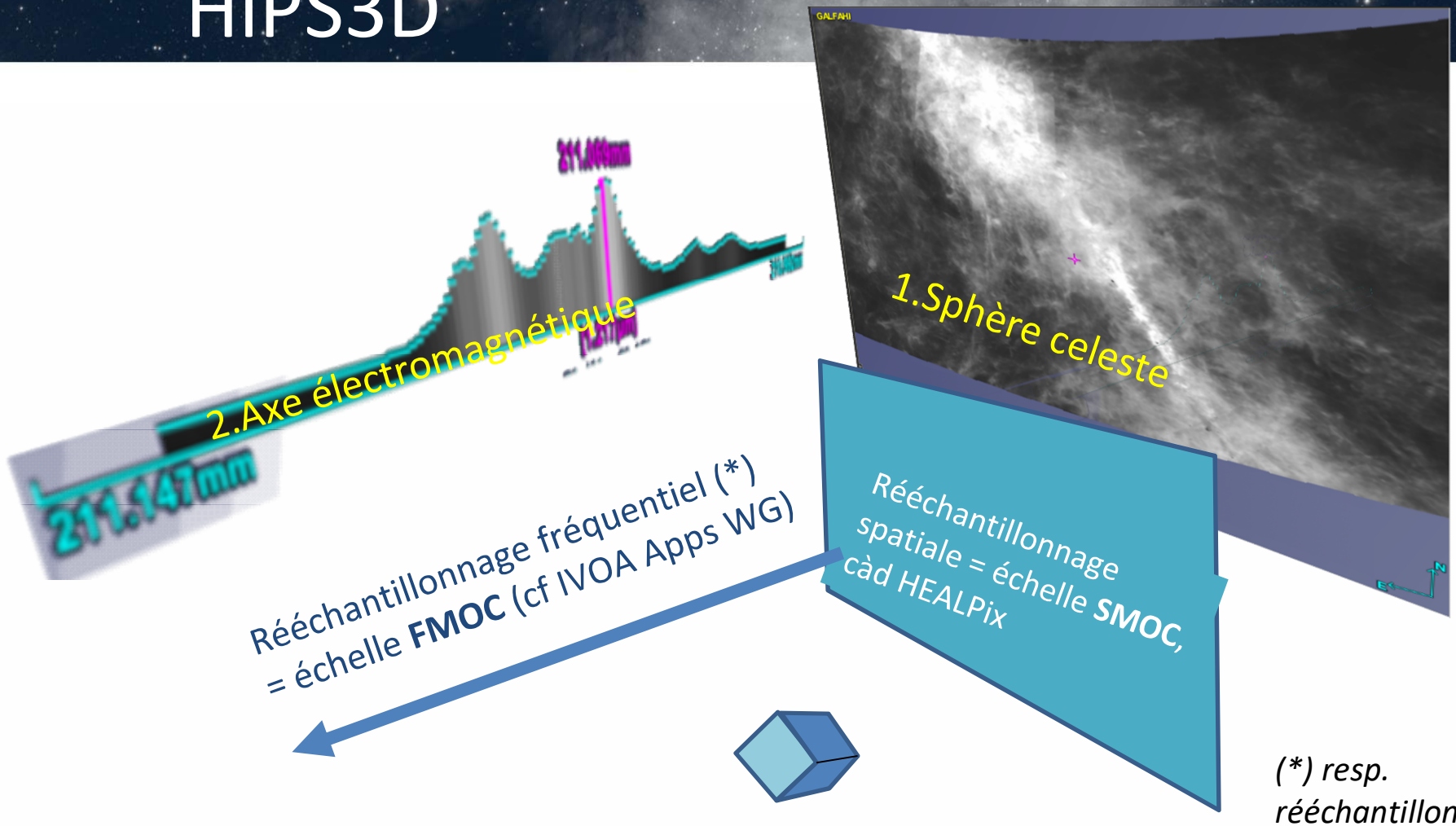
Proposition de solution : HiPS3D(*)

« *Pousser l'idée du HiPS cube actuel vers un "vrai"* ***HiPS 3D*** *càd* ***hiérarchique*** *dans chaque dimension physique et basé sur des* ***unités*** *et un* ***référentiel physique absolu*** »

En d'autres termes : **discrétiser** non pas uniquement la surface de la **sphère**, mais également une **épaisseur**, et cela à plusieurs résolutions.

(*) HiPS3D déjà brièvement présenté à l'Interop + ADASS Malte 2024

Principe du rééchantillonnage HiPS3D



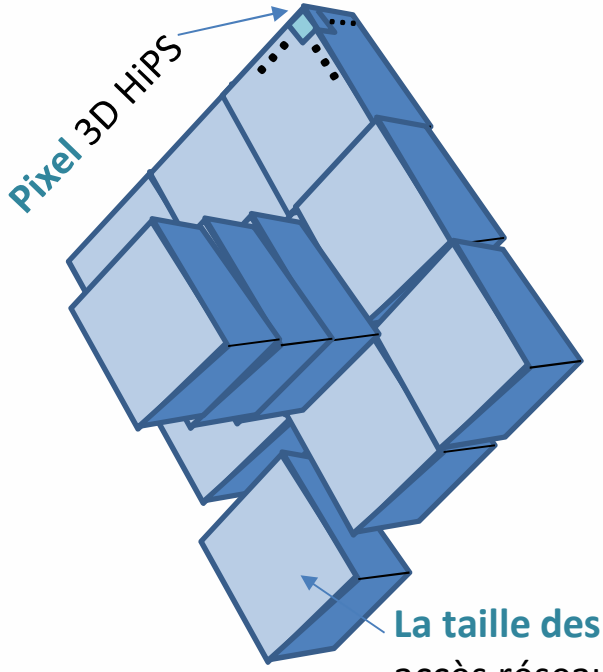
Chaque pixel 3D possède une adresse HiPS3D unique
= (OrderS/NpixS, OrderF/NpixF)

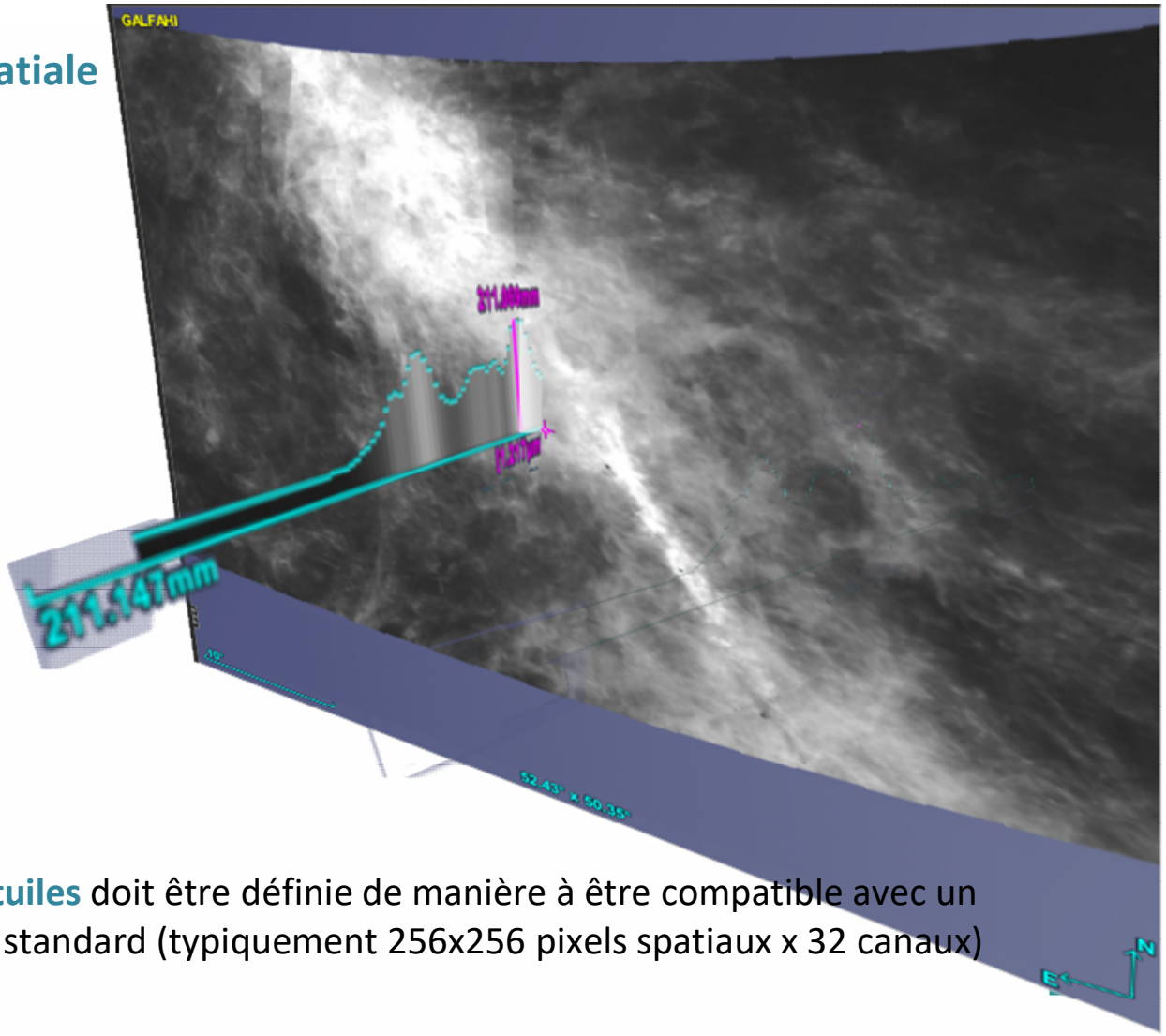
*(*) resp. rééchantillonnage temporel basé sur l'échelle **TMOC***

Principe de « pavage » des tuiles HiPS3D

Le client HiPS3D charge :

- les tuiles couvrant la **vue spatiale**
- les tuiles couvrant la **vue fréquentielle**
- à la **résolution appropriée**





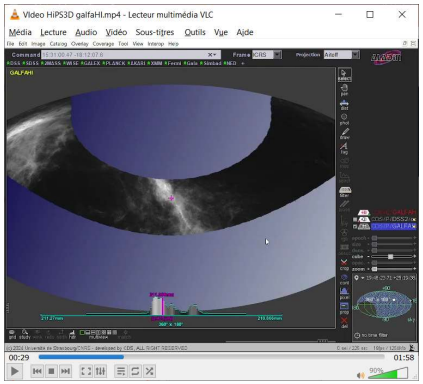
La taille des tuiles doit être définie de manière à être compatible avec un accès réseau standard (typiquement 256x256 pixels spatiaux x 32 canaux)

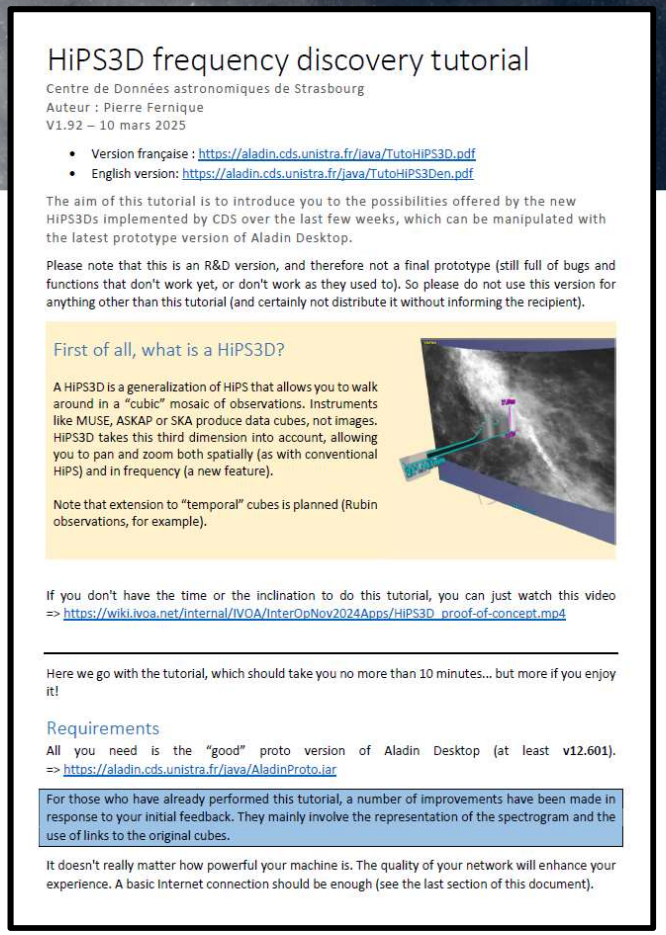
Tests d'implémentation

Développements « **proof-of-concept** » débutés en juin 2024
Orientés **données spatio-fréquentielles** (pas encore sur les données spatio-temporelles)

1. Générateur de HiPS **« Hipsgen »** étendu *(paramètre –hips3d)*
   - Prend en compte des cubes FITS en **fréquences**, **longueurs d'ondes** ou en **vitesse ;**
   - Testé avec succès sur les jeux de cubes :
     - **MUSE** : 2600 cubes (350x350x3700 = 4To) - obs. pointées
     - **SKACD2** : 1 cube (5851x5851x6668 = 850Go) - simulation
     - **ASKAP** : 4 cubes (11000x11000x144 = 177Go) - obs. pointées
     - **GALFAHI** : 225 cubes (512x512x2048 = 225Go) – mosaïque spatiale
     - **MEERKAT**: 3 cubes (5000x5000x2000 = 310Go) – « mosaïque » fréquentielle
2. Client HiPS **« Aladin Desktop »** étendu aux HiPS3D *(version proto 12.6)*
3. Outil coté serveur « **Hips** » dédié à l'extraction dynamique d'un spectre haute résolution à partir d'un HiPS3D (fonction similaire à « Hips2fits »)

Démo/tutorial HiPS3D





HiPS3D frequency discovery tutorial

Centre de Données astronomiques de Strasbourg
Auteur : Pierre Fernique
V1.92 – 10 mars 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 full of bugs and functions that don't work yet, or don't work as they used to). 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://wiki.ivoa.net/internal/IVOA/InterOpNov2024Apps/HiPS3D_proof-of-concept.mp4

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 **v12.601**).
=> https://aladin.cds.unistra.fr/java/AladinProto.jar

For those who have already performed this tutorial, a number of improvements have been made in response to your initial feedback. They mainly involve the representation of the spectrogram and the use of links to the original cubes.

It doesn't really matter how powerful your machine is. The quality of your network will enhance your experience. A basic Internet connection should be enough (see the last section of this document).

https://aladin.cds.unistra.fr/java/TutoHiPS3D.pdf

Proposition d'évolution du standard HiPS:
L'API d'accès aux tuiles

- **Garder la même idée que pour les HiPS existants**, en ajoutant uniquement les suffixes « _x » associés à la dimension additionnelle
- Toujours possible d'utiliser directement un **File System classique** pour le stockage, et un **serveur HTTP** pour la distribution

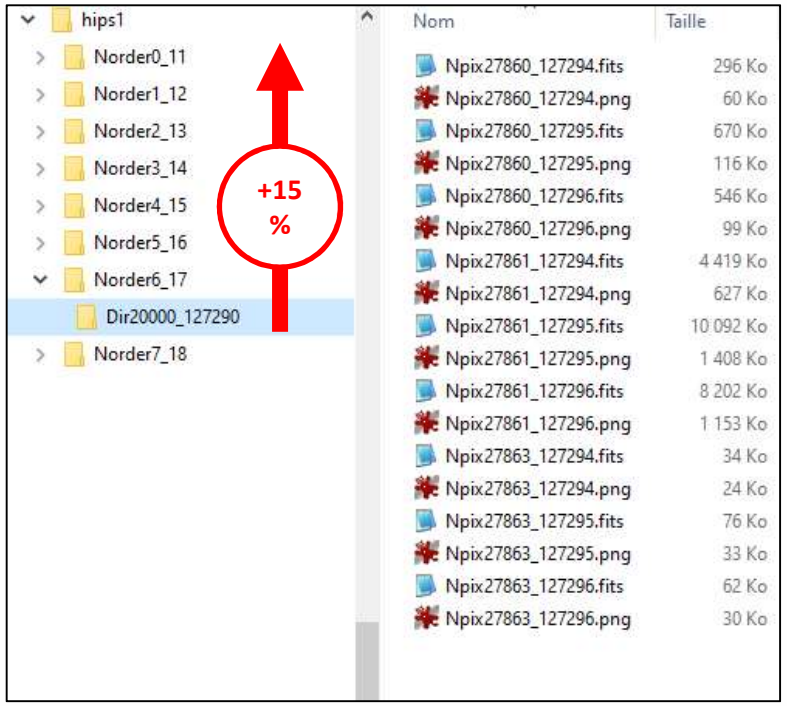
http[s]://server/path/…/Norder***K_L***/Dir***D_E***/Npix***N_M***.*ext*

- **K** est l'ordre spatial, **L** l'ordre fréquentiel,
- **N** l'indice spatial *(méthode SMOC = HEALPix),*
  M l'indice fréquentiel *(méthode FMOC – cf annexe).*
- **D** = (N/10000)*10000, **E**= (M/10)*10 *(division entière)*

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

Structure du répertoire HiPS3D

- Principe: **permettre toutes les combinaisons de résolution** (spatiale vs fréquentielle)(*)
- Mais l'intuition qu'une seule combinaison suffira = **réduction simultanée de la résolution spatiale et de la résolution de fréquence**(*)

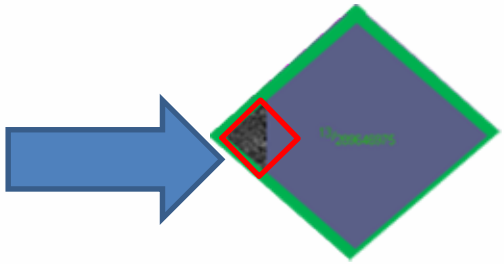


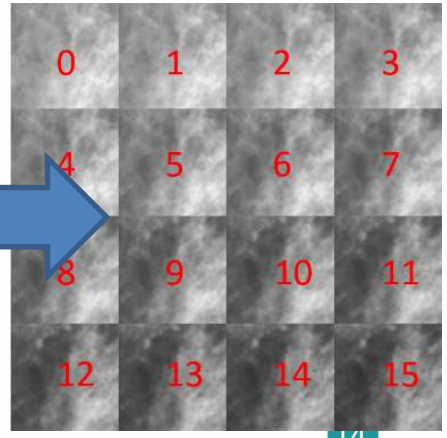
Dès lors, la hiérarchie **n'ajoute que 15 %** au volume de HiPS (réduction d'un facteur 8 pour chaque sous-ordre).

(*) resp. spatiale vs temporelle

Formats des tuiles HiPS3D

- Toujours des **fichiers classiques**
- Mais des **tuiles cubiques**
- Garder l'idée de **plusieurs jeux de tuiles** possibles en fonction de l'utilisation :
  - Dynamique complète => **Tuiles FITS**
    proposition méthode **TRIM** *(suppression des marges vides)* pour réduire l'effet « tuile creuse »,
    voire compression sans perte (ex: RICE)
  - Visualisation rapide => **Tuiles compressées**
    proposition d'usage de compresseurs classiques (jpeg, png, webp…) en **mode damier** (idée T.Boch)
    A noter: compresseurs vidéos décevants (plus lents (x5-10), pas toujours de canal alpha)



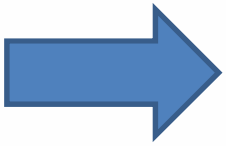


Evolution des métadonnées

- Ajout dans le fichier « **properties** » HiPS
  des 3 mots-clés nécessaires à la nouvelle dimension :
  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
…
```

- Utiliser un **SFMOC**(*) pour décrire la couverture spatio-fréquentielle



Standardiser les FMOC et les SFMOC
(cf présentation Interop IVOA 2024)

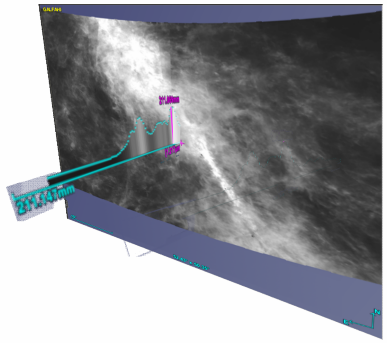
- A minima, imposer les mots-clés **em_min, em_max**(*)

(*) resp. STMOC, t_min, t_max

Prochaines étapes ?

Plus de tests, plus de **testeurs** => essayez le tuto svp
Support client **Aladin Lite** => dans les tuyaux

Démarrer une **nouvelle version IVOA HiPS** => **2.0** ?
(in fine, pas beaucoup de changements)



Merci à toutes les personnes déjà impliquées!
Merci au projet SKA qui suscite
tous ces remue-méninges

Annexe 1: Vocabulaire (proposition)

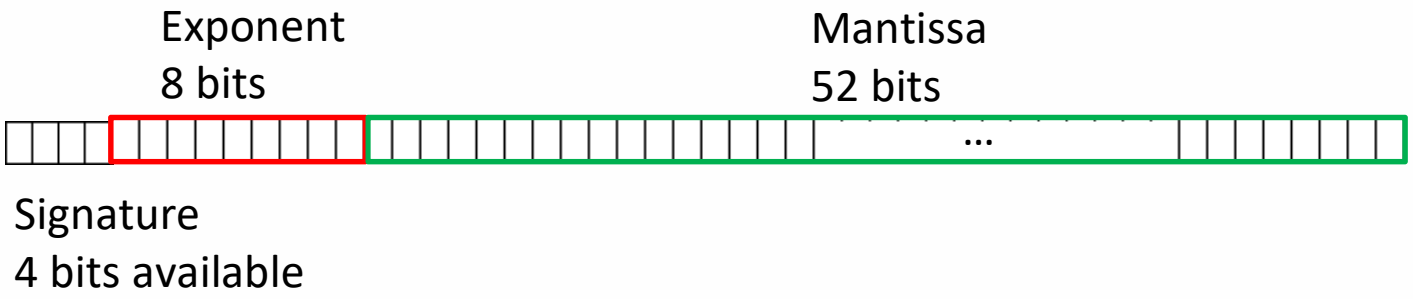
- **HiPS** : Partitionnement hiérarchique de la surface d'une sphère (basé sur une discrétisation SMOC = HEALPix)
- **HiPS3D** : Partitionnement hiérarchique d'une sphère dont la surface a une épaisseur
  - **HiPS3D-time** : l'épaisseur est partitionnée via une discrétisation linéaire (échelle TMOC)
  - **HiPS3D-freq** : idem mais avec une méthode logarithmique (échelle FMOC)
- **pixel** : élément minimal du partitionnement HiPS
  - Cas spatial : losange HEALPix (surface constante pour un order donné)
  - Cas temporel : intervalle de temps (durée constance pour un order donné)
  - Cas fréquentiel : Intervalle de fréquences (intervalle logarithmique pour un order donnée)
- **tile** : regroupement de pixels HiPS « adjacents » (suivant règle HiPS)
  - **width** : nombre de pixels linéairement regroupés (en temporel et fréquentiel), et nombre de pixels d'un côté (en spatial)
  - **widthZ** : spécifiquement pour la dimension fréquentielle ou temporelle en HiPS3D
  - **x[,y]** : coord. cartésienne d'un pixel dans une tuile
  - **channel** : numéro du canal d'une tuile épaisse
- **address** : Numéro composite permettant le repérage d'un pixel HiPS, ou d'une tuile HiPS. Il se décompose en 2 valeurs : l'order et le npix
  - **order** : l'ordre HiPS
  - **npix** : l'index du pixel dans l'order spécifié
    En cas de HiPS3D, 2 valeurs supplémentaires seront ajoutées à l'adresse
  - **orderZ** : l'ordre fréquentiel (respectivement temporel)
  - **npixZ** : l'index fréquentiel du pixel (respectivement temporel)

    L'écriture ASCII de l'adresse suit la syntaxe : **order/npix** et **order/npix_orderZ/npixZ** pour les HiPS3D.

Annexe 2: Discrétisation fréquentielle
(Idée: 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



Déjà présenté à l'Interop 2024 pour une extension fréquentielle des MOC

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;
  }
```