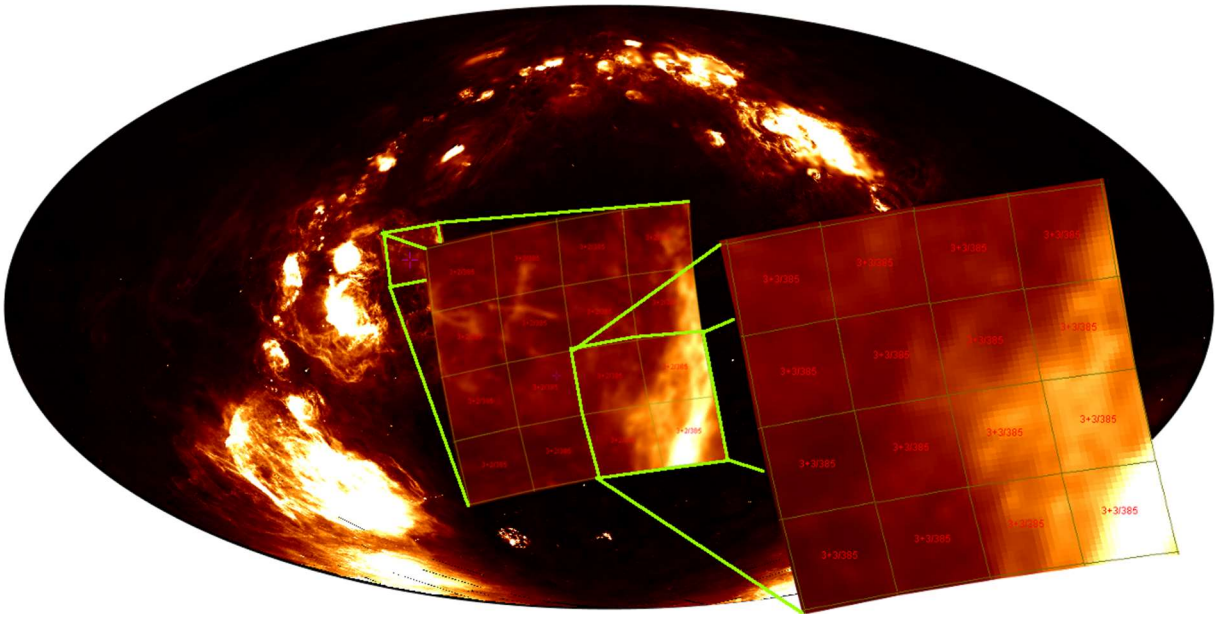
Hipsgen

User manual

Related to versions 12.135 and following – includes HiPS3D.
Last additions appear in green, and in brown.



Translation

with help from deepl.com

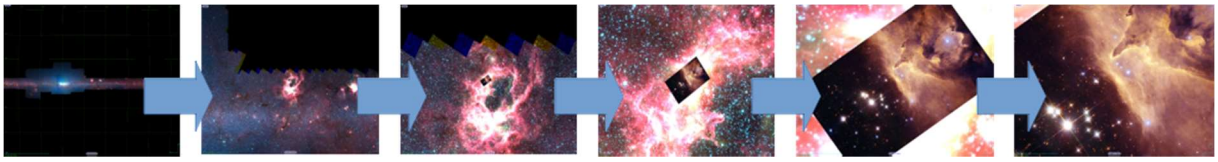
Hipsgen – User manual
Pierre Fernique

January 2023 – with additions 26/1/2024, 23/4, 27/5/2024, 25/2/2025, 30/6 29/9 and 26/2/26

Centre de Données astronomiques de Strasbourg
Observatoire astronomique de Strasbourg

Introduction

Hipsgen is a software package dedicated to generating a Hierarchical Progressive Survey, or HiPS, from a set of images, also known as a "sky survey". A HiPS is both a format and a protocol for viewing astronomical data, based on a hierarchical "tiling" that allows zooming and moving through astronomical images represented as a coherent entity. It is defined in an international IVOA standard "HiPS 1.0".



HiPS is primarily dedicated to scientific use. The methods adopted minimise spatial distortions and, if necessary, preserve the dynamic range (pixel values) of the original images.

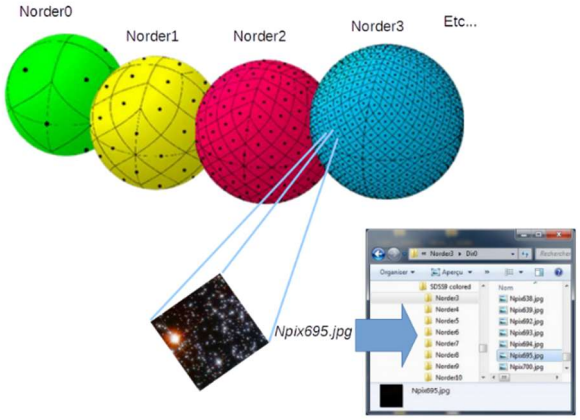
Once the HiPS is created, it will be distributable by a standard web server (Apache, nginx or equivalent) and viewable by any HiPS-compatible astronomical tool such as Aladin (Desktop or Lite), WWT, Stellarium or DIGISTAR, either locally or over the Internet.

Hipsgen is a software developed by the CDS. It shares the same code with Aladin Desktop. It can be used either via the Aladin graphical interface via the "Tools -> Generate a HiPS based on" menu, or from a command line. This document details the command line usage which offers more controls and is more adapted to the production of large HiPS. If you wish to use the graphical interface, please refer to the Aladin Desktop user manual[1].

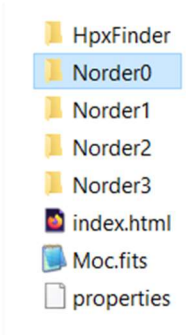
You can download the latest version of Hipsgen at https://aladin.cds.unistra.fr/java/Hipsgen.jar , or simply use the Aladin Desktop code as explained below.

[1] https://aladin.cds.unistra.fr/java/AladinManual.pdf

HiPS structure



In short, a HiPS, or Hierarchical Progressive Survey, is a mosaic of astronomical images stored as hierarchical tiles using the HEALPix sky tessellation. The creation of a HiPS consists of the generation of the mosaic, the partitioning, and then the generation of the tree of tiles that constitute the final HiPS. The resulting tiles are image files of the same size grouped in directories following a hierarchy described in the IVOA standard.



In addition to the tiles, a HiPS provides some additional files, optional or not: index.html - an HTML page allowing to view the HiPS through the Aladin Lite web client, Moc.fits - a file describing the spatial coverage of the HiPS, and properties - a text file providing the characteristics of the HiPS (name, identifier, description, technical characteristics, etc). Finally, you will often find a HpxFinder directory. It is added by Hipsgen to store the spatial index needed to build the HiPS as well as to provide some advanced features.

For more details, please refer to the HiPS IVOA standard [2]

How Hipsgen works

Hipsgen offers a wide range of actions related to the generation and manipulation of a HiPS: spatial indexing, generation of different HiPS tile formats. These actions can be performed separately or automatically one by one. Hipsgen also allows to perform complementary operations such as duplication, concatenation of 2 HiPS, or compliant test on HiPS already created.

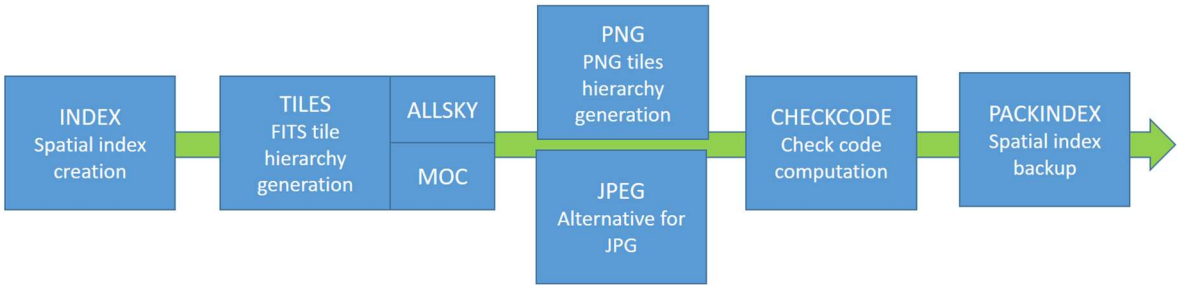
The generation of a HiPS usually follows the following 5 steps:

1. **INDEX:** Generation of a spatial index. This first step allows Hipsgen to memorize the HiPS tiles concerned by each original image (also called "progenitor"). This location information will be stored in the dedicated directory: "*HpxFinder*".
2. **TILES**: Generation of HiPS tiles. Based on the location information obtained in the previous step, Hipsgen will calculate the HiPS tiles one by one in a format that preserves the pixel dynamics of the original images. These tiles are called "FITS" and have the usual size of 512x512 pixels. This step generates not only all tiles with the highest resolution (Norder***x*** where *x* is an integer corresponding to the maximum depth/resolution of the survey), but also all tiles in the HiPS hierarchy up to the lowest resolution (Norder***0***).
   At the end of this step, Hipsgen will also update the files: "*properties*", "*Moc.fits*", and if necessary "*Allsky.fits*" (see IVOA HiPS standard).
3. **PNG**: Generation of a second set of HiPS tiles in PNG compressed image format. This step uses the FITS tiles calculated in the previous step.
   At the end of this step, Hipsgen will update the "*properties*" files and if necessary "*Allsky.png*".

[2] https://www.ivoa.net/documents/HiPS/

This step can be substituted by JPEG which performs the same operation but in a JPEG compressed image format.

4. **CHECKCODE**: Compute the numeric keys. This step will update the properties file with the numerical key values which will allow to quickly check if the HiPS has been damaged, for example after a transfer.
5. **PACKINDEX**: Rewrites the spatial indexing information created in the first step (INDEX) in a binary backup format.



Implementation

Syntax: Hipsgen, on the command line, follows this syntax:

```
java -jar Hipsgen.jar -options in=dirSrc out=dirDst otherParam… ACTIONs …
```

The source directory indicated by the "in" parameter contains the original images. This directory can itself be subdivided into subdirectories, Hipsgen will take into account all the images included in this directory and subdirectories. Alternatively, you can specify an ASCII file that contains the list of paths to the relevant images[3]. Note that if you have only one original image, you can directly indicate its file name.

The destination directory indicated by the "out" parameter will hold the HiPS that will be generated.

Actions: They are indicated as one or more keywords giving the sequence of steps to be performed. By convention, they are written in uppercase even if Hipsgen does not make any distinction. If no actions are mentioned, the 5 actions indicated above (INDEX, TILES, PNG, CHECKCODE, PACKINDEX[4]) will be executed in sequence.

Parameters: They specify some elements required to perform the actions. They follow the syntax "parameterName=values...". As for the actions, the upper and lower case of the parameter name do not matter. Some parameters support aliases (synonyms) for compatibility with previous versions of Hipsgen and/or to take into account the specific IVOA obscore equivalent vocabulary (see section "Metadata").

If the list of parameters is getting longer, the solution of queuing them all up in the command line can be tedious, especially if your parameters contain spaces that require you to enclose them in quotes. You can group some or all of the parameters in a file that will be taken into account by Hipsgen using the "-param=*filename*" option. This file

```
1 in=/data/SCUBA
2 out=/data/ScubaHips
3 id=CDS/P/Scuba/Test
4 blank=0
```

[3] One image path per line; blank lines and lines beginning with the # character are ignored.

[4] Note: The "DETAILS" action is no longer executed by default on Hipsgen versions later than 12.641. It will therefore be necessary to execute it explicitly if access to progenitors is desired (see "DETAILS" action).

must contain one parameter per line, following the syntax "key=value".[5] Note that actions and options (e.g. "-color") should not be put in this parameter file. If they are, they will be ignored.

```
java -jar Hipsgen.jar -param=/dir/param.txt …
```

Options: They control the general behavior of Hipsgen, (e.g. -color: coloring of messages, -n: effective non-execution of actions, etc.). They are preceded by a dash.

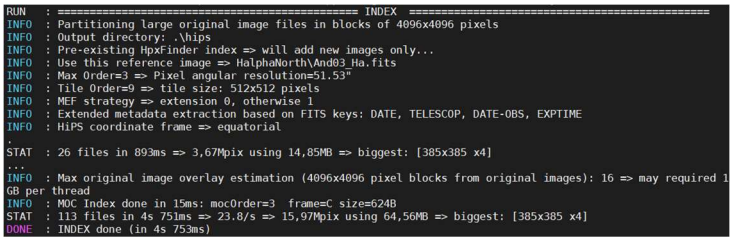
Online help: Hipsgen provides online help for all actions, parameters and options. The exhaustive list of these items can be obtained with the "-h" option (reproduced at the end of this document). The specific man page for one of them can be obtained with the "-man *xxx*" option, where *xxx* is an action or a parameter. Otherwise, all the man pages will be returned.

```
E.g.: java -jar Hipsgen.jar -man TILES
```

Launch of Hipsgen via Aladin.jar: Note that it is also possible to launch Hipsgen with the Aladin Desktop code by using the following command:

```
java -jar Aladin.jar -hipsgen …
```

Terminal: Hipsgen provides numerous indications when executing actions. To help you, and when the terminal allows it, Hipsgen uses a colour code to make it easier to locate information, statistics, alerts and even errors. You can disable this mechanism with the "-nocolor" option or force it with "-color" if Hipsgen has not correctly detected a "colourable" terminal…



Visualisation, distribution and publication

Once your HiPS has been generated, you can display it directly from the "index.html" cover page using a simple Web browser, either locally or remotely via the Internet. In the second case, you will first have to "install" the HiPS you have generated on your web server (apache, nginx or equivalent). As a HiPS generated by Hipsgen[6] is only composed of files, a simple copy in a directory accessible by your web server will work perfectly well without modifying the web server configuration.

To publish your HiPS, in particular if you want your HiPS to appear in the menus of tools such as Aladin or DIGISTAR, please refer to the HiPS standard mentioned above[7].

Generating a HiPS from FITS images

You have the full set of FITS images from a sky survey and you want to generate a HiPS. This is the most common use of Hipsgen. This set of images can be a single image, typically a full Cartesian sky map, or thousands or even millions of images for a large survey such as PanSTARR.

[5] Blank lines and comment lines starting with the # character are allowed.
[6] Note that HiPS can be implemented in other forms (databases, dedicated files) as long as they can be accessed according to the IVOA HiPS standard.
[7] https://www.ivoa.net/documents/HiPS

Prerequisites: All your images must be located in the 'in' directory (or sub-directories) mentioned in the Hipsgen command. These locations can be provided by regular file names or via one or more "symbolic links", or even a network mount, but keeping in mind that fast disk access to these images is strongly recommended. All these images must have an astrometric calibration. These are keywords contained in each FITS image header that allow you to know the position of each pixel on the sky, and vice versa. To ensure that your images contain this information, and in a syntax supported by Hipsgen, just load one of them into Aladin Desktop and check if it is possible to display the coordinate grid. Hipsgen supports most common astrometric calibrations.

The Hipsgen command: Generating a HiPS from your FITS image set consists of simply indicating the directory of your images "in=*xxx*" and the directory where your HiPS will be generated "out=*xxx* and an identifier for your HiPS. This is the minimum required to generate a HiPS.

```
java -jar Hipsgen.jar in=/data/img out=/data/hips id=HERE/P/myhips[8]
```

Identification: The identification of a HiPS is mandatory. It is performed by the parameter "id=...". The identifier follows the HiPS convention of "*ivo://AUTHORITY/P/xxx*". The prefix *ivo://* can be omitted, *AUTHORITY* is a word or an abbreviation - usually in capital letters - of your home institute, ***P*** indicates that it is a HiPS "***P***ixels"[9] , and *xxx* is the specific label of your HiPS. This label can itself incorporate '/' to describe sub-categories of the same mission.

For example, the colour DSS HiPS generated by CDS has the identifier "*CDS/P/DSScolor*", the HiPS of the XMM mission, EPIC instrument generated by ESA has the identifier "*ESAVO/P/XMM/EPIC*"[10].

```
E.g.: in=… out=… id=HERE/P/myhips …
```

Note that you can also modify the identifier of your HiPS by editing the "properties" file afterwards (see the section "Metadata of a HiPS").

Compression: If your images are compressed, either externally by GZIP, BZIP2 or ZIP, or internally by RICE, GZIP1, GZIP2 or HCOMPRESS compression, you don't need to decompress them beforehand, Hipsgen will do it progressively, saving you the time and disk space needed to decompress the entire survey.

Multi-extensions : Your FITS files can contain one or more extensions, (HDU in FITS terminology). By default Hipsgen takes into account the first image extension. The parameter "hdu=n1,n2-n3,...|all" allows you to change this default behaviour. You can explicitly specify the index or indices, "label "or "labels" of the required image extensions, or even take into account all image extensions. In the case of "labels," these can be explicit (e.g., hdu=image) or use wildcards "*" and "?" (e.g., hdu=img*,expos*). In this case, Hipsgen uses the content of the FITS keyword "EXTNAME" to check whether the extension should be taken into account or not.

Additional standard parameters

The FITS format for astronomical images uses some specific parameters that Hipsgen may have to take into account. These parameters may concern the adjustment of the number of bits used to encode each pixel (BITPIX), or the use of a specific null value (BLANK) or a possible linear change to be applied to the pixel values (BZERO, BSCALE).

---

[8] In the remainder of this manual, for reasons of conciseness, this parameter may be omitted from the examples.
[9] There are other types of HiPS, for example **C**atalogue HiPS.
[10] List of currently known HiPS identifiers => https://aladin.cds.unistra.fr/hips/list

Blank: The "blank=*nnnn*" parameter is used to indicate to Hipsgen a specific value used in the original images for the pixels considered as null (i.e. not defined). This value should have been specified directly in each FITS file using the FITS keyword "BLANK", but is often forgotten. The numerical value *nnnn* is the value stored in the FITS file without applying a linear change (bzero/bscale see below). In the case of a FITS image coded in real numbers (BITPIX=-32 or BITPIX=-64), the NaN value will always be considered null even if a specific blank has been mentioned[11]. If *nnnn* is not a numeric value, it will be considered as an alternative to the FITS keyword "BLANK"[12].

E.g. : … `in="…" out="…" blank=0`

Bitpix: The "bitpix=*nn*" parameter is used to modify the coding of the pixels. It indicates the encoding used for FITS tiles whatever the encoding of the original images is. It follows the FITS standard, i.e. 8, 16, 32 or 64 for an integer encoding on the specified number of bits, and -32 or -64 for a real encoding in 32 or 64 bits. By default, Hipsgen keeps the original encoding, or at least the encoding of the first image processed ("reference image").

In the case of a conversion, the range of values may be reduced, which may introduce rounding. To control this effect, the parameter "dataRange=min max" allows to indicate to Hipsgen the range of original values to be taken into account. Hipsgen will then adjust the conversion of the values using the linear conversion operation specific to the FITS standard, i.e. introduce a linear conversion using the BZERO and BSCALE factors according to the formula: *pixval = bzero + bscale x pixcoded*. Note that the original images may have already used a linear conversion by means of their own BZERO and BSCALE, the min and max bounds of the "dataRange" parameter are then to be considered as the pixel values after applying the linear conversion (and not the value encoded in the original FITS file). If this range is not explicitly indicated, Hipsgen will evaluate the content of the first image to be processed ("reference image" - see the "img=xxx" parameter described below) in order to determine the values.

E.g. : `in="…" out="…" bitpix=16 "dataRange=-11000 32000"`

Note that the original images do not have to share all the same encodings. This is true for the BITPIX, but also for the 3 other parameters BLANK, BZERO and BSCALE.

8 bits compressed tiles: Hipsgen will generate FITS tiles and PNG compressed tiles. For PNG tiles, the pixels are encoded on 8 bits reducing the dynamic range of possible values. Therefore, similar to a BITPIX conversion in the case of FITS tiles, the "pixelCut=min max [fct]" parameter will allow you to explicitly indicate the range of original values to be taken into account and the "transfer function" to be applied to obtain the 255 possible values of the compressed tiles: log, sqrt, linear (default), asinh, pow2. If this interval is not mentioned, Hipgen will scan the first image ("reference image") to be processed and will automatically determine the range that seems appropriate. Note that the values considered as null (cf blank above) will appear transparent in PNG compressed tiles, and in black for the JPEG alternative (cf JPEG action).

E.g. : `in="…" out="…" pixelCut="-8000 16000 sqrt"`

Determining a range of values for the generation of 8-bit images can be difficult, especially when the sky survey contains original images with a very high amplitude of values. It is necessary to find the best compromise in order to avoid generating a HiPS in which some compressed tiles are too dark and

[11] The use of a FITS BLANK keyword for a real -32 or -64 encoding is surprising, but not forbidden.
[12] The use of an alternative FITS keyword to the conventional word BLANK is quite rare. For example, "BADVAL" can be found.

others too light. An alternative solution exists for pointed image surveys. Since these surveys do not form a continuous mosaic on the sky, it may be possible to determine a specific interval for each observed area. With the parameter "pixelCut=byRegion..." Hipsgen will determine independent observed regions, evaluate for each of them a specific interval by analysing the pixel distribution of the concerned FITS tiles. Then it will save these thresholds in the header of the FITS tiles using the keywords "CUTMIN" and "CUTMAX". These values will be used when generating the compressed tiles. Thus, the resulting HiPS will be much more visually homogeneous.

When Hipsgen performs an automatic evaluation to determine a range of values, either on the reference image or on the independent regions, the pixelCut parameter can be used to provide two percentages indicating the proportion of pixel values to be rendered. Thus the example below asks Hipsgen to determine the minimum and maximum thresholds for rendering between 3% and 98.5% of the pixel value distribution for each independent region of the survey.

E.g.: `in="…" out="…" pixelCut="3% 98.5% byRegion"`

Sky background: Depending on the origin of the images, the sky background may or may not have already been subtracted. Therefore, it may be necessary to perform a "sky background" adjustment to be able to make a mosaic without a "patchwork" effect. Sometimes, the sky background value has not been subtracted, but simply indicated in the header of each image by means of a dedicated keyword (e.g. BACKGRD). The "skyVal=key" parameter allows you to tell Hipsgen which keyword should be used to subtract the sky background. This parameter can also be set to: "auto" - indicates to Hipsgen to make an evaluation of the sky background on each image[13], "info%" - represents the centered percentage of the automatic sky background detection histogram that Hipsgen should keep, or "min% max%" - represents the range of percentages of this histogram.

E.g.1 : `in="…" out="…" skyVal=BACKGRD`
E.g.2 : `in="…" out="…" skyVal=99%`
E.g.3 : `in="…" out="…" skyVal="0.3% 99.7%"`

Exposure time: The original images may also have a specific exposure time. In order to obtain a homogeneous mosaic, and in case this information has been mentioned in the FITS header of each original image, it may be necessary to ask Hipsgen to average pixel values as a function of exposure time. The parameter "expTime=*fitskey*" is used to indicate the relevant FITS keyword[14].

E.g. : `in="…" out="…" expTime=EXPTIME`

Reference image: When Hipsgen processes a set of images, it will use the characteristics of one of them (the first one during the sequential reading of the input directory) and deduce from it, among other things, the bitpix, order, dataRange, pixelCut parameters to apply. By using the img=*filename* parameter, it is possible to explicitly indicate which "representative" image should be used.

E.g. : `in=/data/img out="…" img=/data/img/Image32.fits`

[13] The evaluation of the sky background by Hipsgen uses 5 zones of the image distributed as on the face of a dice, and takes into account the averages of the measurements on the 3 zones whose values are not the extrema.
[14] This method is not restricted to the exposure time, it can be used for any value of the FITS header as weight factor.

Resampling: HiPS is based on a division of the sky into HEALPix, i.e. 12 diamonds of identical spherical surfaces at the lowest order, each subdivided into 4 at each additional order. The maximum resolution is obtained at the 29th subdivision and corresponds to an angular size of about 400 µas per diamond. HiPS tiles group together a square of NxN HEALPix diamonds, by default 512x512 (N is necessarily a power of 2). The "tileWidth=*nnn*" parameter allows to modify this default, either to make smaller tiles (e.g. 64x64) which will reduce the "null" edges in the case of pointed observations, or to make larger tiles (e.g. 1024x1024) to reduce the total number of HiPS tiles. Knowing that the tile is the basic element that a HiPS client loads to perform the visualisation, it is important to keep the size "reasonable" in terms of KB to transfer/handle.

k	$N_{side} = 2^k$	N_{pix}	θ_{pix}	$k_{tile,512}$	$N_{tile,512}$	$\theta_{tile,512}$
0	1	12	58°.6			
1	2	48	29°.3			
2	4	192	14°.7			
3	8	768	7°.33			
4	16	3072	3°.66			
5	32	12,288	1°.83			
6	64	49,152	55′.0			
7	128	196,608	27′.5			
8	256	786,432	13′.7			
9	512	3,145,728	6′.87	0	12	58°.6
10	1024	12,582,912	3′.44	1	48	29°.3
11	2048	50,331,648	1′.72	2	192	14°.7
12	4096	201,326,592	51″.5	3	768	7°.33
13	8192	805,306,368	25″.8	4	3072	3°.66
14	2^{14}	3.22×10^{9}	12″.9	5	12288	1°.83
15	2^{15}	1.29×10^{10}	6″.44	6	49152	55′.0
16	2^{16}	5.15×10^{10}	3″.22	7	196608	27′.5
17	2^{17}	2.06×10^{11}	1″.61	8	786432	13′.7
18	2^{18}	8.25×10^{11}	0″.81	9	3,145,728	6′.87
19	2^{19}	3.30×10^{12}	0″.40	10	12,582,912	3′.44
20	2^{20}	1.32×10^{13}	0″.20	11	50,331,648	1′.72
21	2^{21}	5.28×10^{13}	0″.10	12	201,326,592	51″.5
22	2^{22}	2.11×10^{14}	50.3 mas	13	805,306,368	25″.8
23	2^{23}	8.44×10^{14}	25.1 mas	14	3.22×10^{9}	12″.9
24	2^{24}	3.38×10^{15}	12.6 mas	15	1.29×10^{10}	6″.44
25	2^{25}	1.35×10^{16}	6.29 mas	16	5.15×10^{10}	3″.22
26	2^{26}	5.40×10^{16}	3.15 mas	17	2.06×10^{11}	1″.61
27	2^{27}	2.16×10^{17}	1.57 mas	18	8.25×10^{11}	0″.81
28	2^{28}	8.65×10^{17}	0.786 mas	19	3.30×10^{12}	0″.40
29	2^{29}	3.46×10^{18}	0.393 mas	20	1.32×10^{13}	0″.20

Hipsgen determines the number of subdivisions required based on the tile size (512x512 by default) and the angular resolution of the original images. It uses the first processed image (standard image). It will automatically choose the necessary order to have an angular resolution of the HiPS pixels equal to or just above the initial resolution. Order 0 (k-tile in the table[15]) corresponds to the 12 starting tiles. Order 9 corresponds to 3,714,728 tiles, and if the tiles are 512x512 pixels in size, they provide an angular resolution per pixel of 805.2 mas. Each additional order is twice more accurate in angular resolution, but adds 4 times the volume of the previous order. By using the "order=*nn*" parameter it is possible to subsample, or increase the sampling that Hipsgen had planned by default.

E.g. : `in="…" out="…" tileWidth=256 order=13`

Hipsgen uses a bilinear resampling algorithm to "transform" the pixels from the original images to the HEALpix grid. Hipsgen does not offer an alternative to this algorithm, unless the original pixels are already located in a HEALPix grid (see the section "Generating a HiPS from a HEALPix map").

Overlay: Hipsgen is used both on image sets pointing to astronomical objects, and on sky surveys in the form of a mosaic of images with partially overlapping edges. In both cases, several original images may contribute by overlap to the pixel values calculated in the HiPS tiles. The "mode=*param*" parameter is used to indicate the method Hisgen should use to combine multiple original pixel values. The supported parameters are: overlayNone - no overlap, only one pixel value will be considered,

[15] Table extracted from the paper Fernique et al (2015A&A...578A.114F). The columns represent: k - the HEALPix order, Nside - the resolution in the HEALPix terminology, Npix - the number of HEALPix pixels, θ-pix - the angular resolution, k-tile - the HiPS tile order, N-tile - the number of HiPS tiles, θ-tile - the angular resolution of a pixel in the tile.

overlayMean - all values will be averaged, overlayMean*X* – The first *X* values will be averaged[16], overlayAdd - all values will be summed.

E.g. : `in="…" out="…" mode=overlayMean`

Note that in the case of averaging (default mode), the signal-to-noise ratio will be a function of the number of source images contributing to the final value. The consequence is that it is normal to be able to detect astronomical objects in the HiPS that were not detectable in each individual image.

In addition, Hipsgen (version 12.134 and higher) provides a COUNT action that generates an alternative HiPS (dedicated HipsCounter directory) counting, for each HiPS pixel, the number of images that contributed to its calculation (see section "Generating a weight HiPS").

Suspicious images: It may happen that in the original set of images, some of them have an imprecise or even incorrect astrometric calibration. The result on HiPS will depend on the nature of the error and usually consists of a slight shift of objects on the sky, easily spotted when overlapping several original images over the same region. But a particularly annoying error is an erroneous angular pixel size. If the error is large these pixels will be able to alter a large area of the HiPS. To avoid this, for a set of several images, Hipsgen automatically rejects all images whose pixels have a suspicious angular projection on the sky because they are strongly disproportionate in longitude compared to latitude. By default a ratio greater than 3 will be considered improbable[17]. The parameter "maxRatio=*x*" allows to modify this limit value, and the value 0 suppresses the test completely. The list of discarded images is provided at the end of the processing.

E.g. : `in="…" out="…" maxRatio=1.5`

Field of view: Some original images may have a field of view smaller than the totality of the pixels of the image: altered edges, the existence of a cartouche, the field of view of the telescope being smaller than the image, ... To ignore the "unobserved" original pixels of such images, Hipsgen can take into account a polygon or a circle where only the internal pixels will be taken into account. The polygon or circle can be the same for all images, or only for part of the images, or even for each image individually. This polygon/circle will be described in a file with the same name as the image, respectively the directory, to which it is associated but with the extension ".fov". In the case of a ".fov" file associated with a directory, all the images in that directory will be concerned, except for those which have their own ".fov" file. For a polygon the expected format is a list of X Y pairs separated by a comma and/or a space, one pair per line, described counterclockwise in the image. For a circle, Hipsgen expects a single line with a triplet representing the X,Y centre and its radius R.

The coordinates are to be used according to the FITS convention, i.e. the centre of the pixel at the bottom left is at the coordinates 1,1. It is essential to use the "fov=true" parameter to ensure that these additional files are taken into account by Hipsgen. In the case of a single shape common to all the images, it is also possible to indicate the coordinates of this shape directly as the value of the parameter (ex : fov=2,2,1000,2,1000,500,4,390).

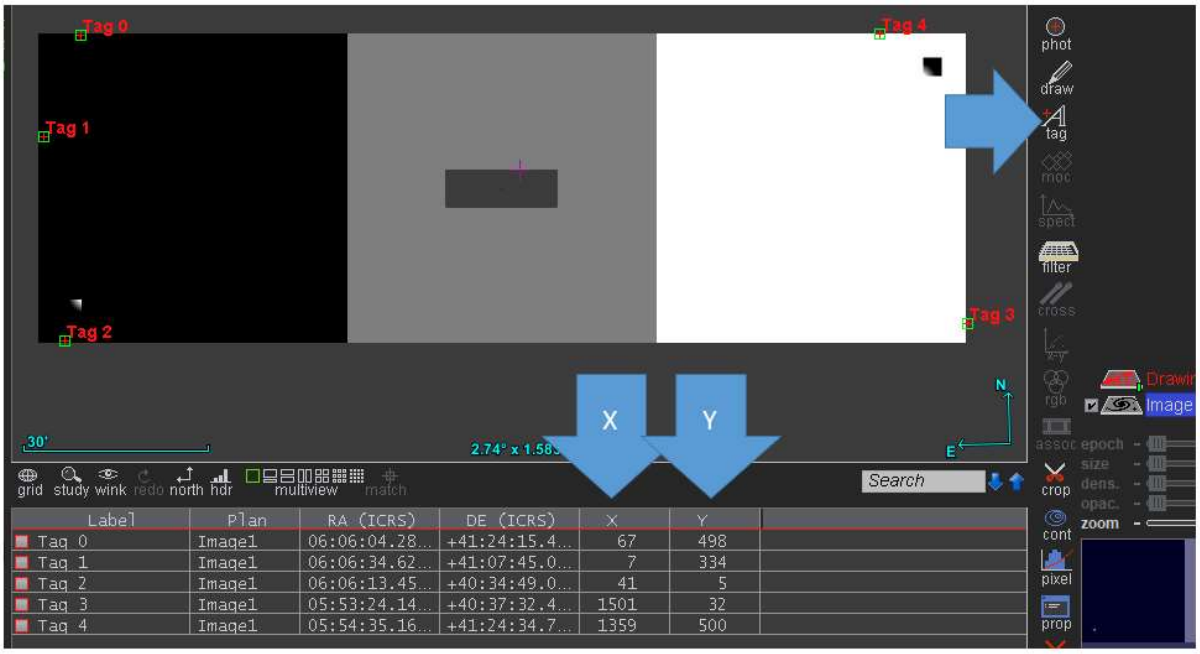
E.g. : `in="…" out="…" fov=true`

---

[16] When using the expTime parameter, the *X* averaged values will automatically be those with the longest exposure times..

[17] Note that this filtering is not applied to images covering a large part of the sky in Cartesian or Moldweide projection.

Tip: *Obtaining X,Y pairs can be made easier via Aladin Desktop by using the "tag" tool.*



When the edges to be removed are constant, the "border=N W S E" parameter allows you to remove a specific number of pixels from the top, left, bottom and right of all images respectively. A single value is also possible if all edges are identical.

E.g. : `in="…" out="…" "border=10 50 20 50"`

When the shape of the field of view is a rectangle or ellipse, aligned in the image, but the centre or even the size is not identical from one image to another, the "shape=rectangle|ellipse" parameter will tell Hipsgen to determine the mask characteristics for each image by itself[18].

E.g. : `in="…" out="…" shape=ellipse`

Hierarchy: During the generation of the tile hierarchy, each tile of order ***x*** is generated from its 4 daughter tiles of order ***x+1***. The "mode=*param*" parameter is used to specify the aggregation method for each pixel: treeMean - the average of the 4 child pixels, treeMedian - the median of the 4 child pixels, treeFirst - one of the 4 pixels, treeMiddle – one the 2 intermediate pixel values. If not specified, Hipsgen uses the mean for HiPS from FITS images, and the median for HiPS from colour images[19] (see "How to generate a HiPS from JPEG images"). As the "mode" parameter is used for several controls, it is possible to specify several values, or even to specify only the end of a parameter value to apply it simultaneously. The 2 examples below are equivalent. They apply the calculation of the average, both for the overlays (see previous paragraph) and for the hierarchy.

E.g.1 : `in="…" out="…" mode="overlayMean treeMean"`
E.g.2 : `in="…" out="…" mode=mean`

[18] This method should only be used in the case where a BLANK value cannot be defined (typically when the edge value - usually zero - is also used in the pixel area to be considered. This situation is actually quite common (e.g. GALEX, etc).

[19] The choice of the median for the colour images makes large structures, nebulosities, filaments, etc. more visible. The mean is the natural method of preserving photometry between scales in the case of surface brightnesses.

Partitioning: In the case of a large survey, and especially if it has very dense overlays, Hipsgen may need a lot of resources to be able to calculate the pixel values of the tiles. To avoid memory overflows, Hipsgen will subdivide the treatment of very large original images into 4096x4096 pixel blocks. The parameter "partitioning=*nnn*|false" allows you to adjust the size of these blocks, or even to remove this option if you have enough RAM. This parameter must be applied at the first step of the HiPS generation, i.e. the spatial index (INDEX).

E.g. : `in="…" out="…" partioning=false`

Coordinate system: By default, Hipsgen generates the HiPS based on the ICRS equatorial coordinate system. Thus the HiPS tiles will be arranged according to the equator and the equatorial poles. The parameter "frame=equatorial|galactic|ecliptic" allows to modify this default behaviour. However, it is preferable to keep the equatorial system as much as possible in order to be able to use some HiPS tools combining several HiPS, with the constraint of a single reference frame.

E.g. : `in="…" out="…" frame=galactic`

Generating a HiPS from compressed colour images

You have a set of colour images (png or jpg) and you want to generate a HiPS.

Prerequisites: Your images must have an astrometric calibration. Hipsgen will look for this information either in the image files directly, in the form of AVM tags[20], or in a WCS header stored in each image file, or separately in a file with the same name but with the extension replaced by ".hhh". To ensure this, simply load one of these images into Aladin Desktop and check that the coordinate grid can be activated. If this is not the case, you will have to 'calibrate' each of these images manually (e.g. via the Aladin Image -> Astrometrical calibration menu), or automatically (e.g. via Astrometry.net).

Coloured tiles: Unlike HiPS based on greyscale FITS images, Hipsgen will directly generate colour tiles corresponding to the original colour images. By default, the tiles produced will use the same compression format as the original images, or more precisely the first original image processed ("reference image"). The "color=png|jpeg" parameter will allow you to modify this default behaviour. The JPEG format is faster to generate, and produces smaller tiles. The PNG format has a transparency channel that allows the HiPS client to simultaneously view a partial PNG HiPS on top of another HiPS.

```
in="/data/img " out="/data/hips" color=jpeg …
```

Overlays: Overlays of the original images can be processed with various methods. The parameter "mode=*xxx*" allows to tell Hipsgen how to handle multiple contributors to a HiPS pixel: overlayMean - average (colour channel by colour channel), overlayAdd - addition (colour channel by colour channel), overlayNone - single contributor. However, these methods are much more suitable for processing the original FITS images where the pixel values represent a physical quantity. A fourth option is possible "mode=overlayFading". It offers an interesting alternative for colour images. Hipsgen will average the contributors but using a weighting that is inversely proportional to the distance to the edge of the corresponding image. A complicated way to describe a "fade-in" between images to smooth out transitions between images.

[20] Astronomy Visualization Metadata - http://virtualastronomy.org : keywords to describe astronomical images for media use.

Hierarchy: When building the tile hierarchy, each tile of order x is generated from its 4 daughter tiles of order x+1. The "mode=*param*" parameter allows to specify the aggregation method of the pixels: treeMean - the average of the 4 child pixels, treeMedian - the median of the 4 child pixels, treeFirst - one of the 4 pixels, treeMiddle - one of the 2 intermediate value pixels. If not specified, Hipsgen uses the median for HiPS from colour images[21]. The calculations are performed on each colour channel (red, green and blue) independently.


treeMean


treeMedian


treeMiddle


treeFirst

Control and performance

Generating a HiPS on a large set of images is a heavy process in terms of disk space, memory resources and the computing capacity you have. The operation consists of potentially reading billions of pixels, performing various calculations on each of them, and rewriting them. There are several parameters that can be used to customize the performance of Hipsgen.

"Not run": First of all, the "-n" option will allow you to check in advance what Hipsgen will do according to the parameters you have given it. Technically, this option inhibits any writing to the disk. There is no risk of any damage even if you have made a mistake in your settings. On the other hand, you will see the technical information of the treatment (see the section "Hipsgen traces" at the end of this document). This is very useful if you have a doubt, especially if the action you want to take modifies an already generated HiPS that you could damage by mistake.

Pilot: The generation of a HiPS often requires a few preliminary tests to adapt the various parameters and to check the result visually (using Aladin Desktop or any other HiPS compatible client). However, the time required to generate a HiPS can be long, making these fine-tuning steps tedious. The "pilot=*nnn*" option allows you to calculate a HiPS "for seeing" only on *nnn* original images. This allows you to adjust your settings until you determine the best options. Then you run the calculation again on all the images, having removed the "pilot=*nnn*" parameter. Note that if you generate the final HiPS in the same destination directory as your pilot and this one only contains a small number of images, it is then judicious to add the "-clean" option which will accelerate the processing by avoiding Hipsgen to have to constantly check each source image if it has already processed or not (see below section "HiPS computation recovery").

```
Pilot1: in=/data/img out=/data/hips pilot=100
Pilot2: -clean in=/data/img out=/data/hips pilot=100 skyval=auto …
Final execution: -clean in=/data/img out=/data/hips …
```

[21] The "treeMiddle" method particularly highlights large structures. The "treeFirst" method emphasises point sources. The effect is cumulative at each zoom level.

The "pilot" stages also allow you to note the performance you obtain. Of course, they depend on the capacity of your computing machine, but also on the options you have chosen for the creation of the HiPS. During the generation of the tiles ("TILES" action), processing speed statistics are displayed every 30s allowing you to compare the time cost for a expecting result. On a small machine (4 cores/4GB), you can expect 500 to 1000 tiles per minute, and up to 5000 to 7000 tiles per minute for a high performance computer (50 cores/128GB). Thus, a complete sky survey such as 2MASS at 800mas HiPS resolution that requires more than 3 million tiles will take about 15 hours of processing on a big computer.

```
.............
STAT  : 610/17977 tiles + 439 nodes in 12s 796ms (3.4%) by 8/8 threads
STAT  : RAM cache: 67 items/300 using 132,73MB/5,28GB freeRAM=7,73GB (opened=67 reused=672 released=0)
.............................
STAT  : 1906/17977 tiles + 1367 nodes in 43s 232ms (10.6% 2554 tiles/mn(=10,64Mpix/s) EndsIn:6m 4s) by 8/8 th
STAT  : RAM cache: 175 items/300 using 346,7MB/5,28GB freeRAM=7,51GB (opened=175 reused=2159 released=0)
.............................
STAT  : 3054+1/17977 tiles + 2161 nodes in 1m 13s (17.0% 2258 tiles/mn(=9,41Mpix/s) EndsIn:6m 106ms) by 8/8 t
STAT  : RAM cache: 282 items/300 using 558,67MB/5,28GB freeRAM=7,25GB (opened=282 reused=3574 released=0)
```

Tip: *Aladin Desktop version 12 and following can display a HiPS even during generation (fTILES step) without even stopping Hipsgen. This is convenient for impatient people, and also to check the result before the end of the processing. As the HiPS tree is being generated, it may be necessary to zoom in to the deepest order tiles to see the partial result. Note that in order to take into account the progress of the calculations, it is necessary to reload the HiPS in a new Aladin Desktop plan.*

Multitasking: Hipsgen speeds up its processing if it can use multiple cores. By default, it will take all available cores on the machine. If this machine needs to perform other tasks simultaneously, the "maxthread=*n*" option will allow you to limit the number of threads used by Hipsgen and consequently the cores used. Note that it is not required to reduce the number of threads to save memory impact (recommended in earlier versions of Hipsgen), Hipsgen will now do it automatically if necessary.

RAM: One of the resources that Hipsgen uses heavily is RAM. Just like the computer cores, the more you give it, the better it will do. Since Hipsgen is a java program, the control of RAM memory is done by the "-Xmx" parameter specific to your java interpreter. A good practice is to count 1GB of RAM for each calculation core. The example below allocates 10 gigabytes.

```
E.g.: java -Xmx10g -jar Hipsgen.jar in=/data/img out=/data/hips …
```

If you have a very powerful machine, you can consider using almost all your RAM memory (be careful not to use more, otherwise you will swap and slow down processing drastically) while inhibiting the block partitioning that Hipsgen performs on large images (>4096x4096). Since you have memory, you might as well take advantage of it. The example below allocates 255 gigabytes.

```
E.g.: java -Xmx255g -jar Hipsgen.jar partioning=false in …
```

On the contrary, if your machine is a small laptop with very little RAM (typically <1GB), you should impose a block size smaller than the default. Hipsgen will be able to generate your HiPS even on a small machine, just be more patient.

```
E.g.: java -Xmx700m -jar Hipsgen.jar partionning=512 in …
```

Disk cache: In case your original images are compressed, Hipsgen will have to decompress them during processing. For this it needs disk space. As Hipsgen will potentially use the same image several times during its processing, it will save time if the decompression operation is not repeated unnecessarily. So the larger the disk cache, the faster the processing will be. For compressed FITS images, the default disk cache is designated by the operating system of your machine. The "cacheSize=*xxx*" parameter

allows you to increase the default size (500GB or less if there is not enough space on the partition concerned). Note that *xxx* is expressed in MB. If the default cache partition is too small, the "cache=*path*" parameter will allow you to specify an alternative directory. Finally, the "cacheRemoveOnExit=false" parameter will allow you to keep the contents of the cache, and therefore the decompressed images in it, in order to accelerate a possible resume of processing. It is up to you to delete this cache manually once the final HiPS has been obtained.

For JPEG or PNG colour images, if they are particularly large (>4096x4096), a disk cache is also used but directly managed by the java graphics libraries. Therefore, thanks to an option of the java interpreter "-Djava.io.tmpdir=*path*" you can designate an alternative cache directory.

```
E.g.: java -Djava.io.tmpdir=/path/to/tmpdir –jar Hipsgen.jar …
```

Tile size reduction: First of all, note that you can already obtain PNG (or JPEG) compressed tiles via the "PNG" (or JPEG) action. You are free to remove the FITS tiles afterwards via the "CLEANFITS" action in order to finally obtain a HiPS that is only intended for viewing, but with a smaller volume. However, this approach is not adequate if you want to keep the FITS tiles and use them for the full dynamics of the pixel values. Hipsgen offers two possibilities to reduce the size of FITS tiles: edge removal and/or compression. These two methods should only be used when strictly necessary. They are not standardized by the IVOA and are still subject to change, and the FITS tiles produced are currently only recognised by Aladin Desktop.

Edge removal: A first technique consists of performing a “pruning” operation along the edges of the FITS tiles. It is interesting on the fragmented HiPS[22]. It has the advantage of being very fast, even almost instantaneous if used during the generation of FITS tiles. It also has the advantage of not inhibit direct access to pixel values[23]. The FITS tiles are still regular FITS images, but with possibly restricted sizes and with the dedicated TRIM1 and TRIM2[24] keywords storing the "original" edges that have been removed, ONAXIS1 et ONAXIS[25] the original dimensions. Edge reduction can be applied immediately when generating FITS tiles if the "-trim" option is specified. It can also be applied, respectively removed afterwards via the "TRIM" and "UNTRIM" actions.

```
E.g.1: -trim in=/data/img out=/data/hips INDEX TILES …
E.g.2: out=/data/hips TRIM
E.g.3: out=/data/hips UNTRIM
```

External GZIP compression: Hipsgen allows you to apply GZIP compression to FITS tiles, either on the fly with the "-gzip" option, or via the "GZIP" action. This action will gzip all FITS tiles and the "Allsky.fits" file if it has been generated[26]. This is a very efficient operation in terms of volume, but it is time consuming, of the same order of magnitude as the generation of the FITS tiles themselves. Conversely, via the GUNZIP action you can unzip FITS tiles that have been gzipped previously..

```
E.g.1: -gzip in=/data/img out=/data/hips INDEX TILES …
E.g.2: out=/data/hips GZIP
```

---

[22] This approach is especially effective on HiPS of pointed objects for which the HiPS tiles are very "spearsed", i.e. they have many null pixels, and mainly on the edges, and for an increasingly large proportion up the HiPS hierarchy.

[23] This property is required for some HiPS tools working "server side", for example to quickly extract from a group of HiPS the values of a SED on a given position.

[24] Respectively XTRIM and YTRIM in versions prior to 12.5

[25] Respectively ZNAXIS1 and ZNAXIS2 in versions prior to 12.5

[26] Note that FITS tiles, whether gzipped or not, will keep their ".fits" extension. It is up to the client to detect a possible compression thanks to the magic code at the beginning of the file, rather than using the file extension.

```
E.g.3: out=/data/hips GUNZIP
```

Internal FITS 4.0 compressions: Finally, the latest version of the FITS standard describes several modes of compression using "internal" FITS compression. These compressions can be implemented on FITS tiles produced by Hipsgen using the "**fpack**[27]" utility in post-processing. In this way, it will be possible to apply RICE[28] compression, which is more efficient than simple external GZIP compression as described in the previous paragraph. This alternative method produces HiPS that complies with the IVOA 1.0 standard. However, tiles compressed in this way are currently only supported by HiPS Aladin Desktop clients version >= 12.61.

HiPS computation recovery

The generation of a HiPS on a very large survey can take several hours or even days. It is therefore not surprising that processing can be interrupted, voluntarily or not.

Recovery after an involuntary interruption

You have been waiting for several hours during the generation of your HiPS. Unfortunately, the calculation has been interrupted by accident. You can of course restart the calculation, without even deleting the tiles already generated. However, you can specify the "mode=keeptile" option, which will avoid regenerating the tiles already generated, which will speed up the processing..

```
in=/data/img out=/data/hips mode=keeptile …
```

Recovery to change the calculation mode

However, if you want to restart the whole calculation from the beginning (for example if you want to change some parameters such as the BITPIX of the HiPS), you will have to delete the first processing, either partially or totally. To do this you have a series of cleanup actions: CLEAN - deletes the whole HiPS except the properties file, CLEANINDEX - deletes the spatial index (HpxFinder), CLEANTILES - deletes all tiles, CLEANFITS - deletes all FITS tiles, CLEANPNG - deletes all PNG tiles, CLEANJPEG - deletes all JPEG tiles, CLEANWEIGHT - deletes all weight tiles (see –live option).

You can also use the "-clean" option which will automatically delete only those items that will be regenerated[29].

```
-clean in=/data/img out=/data/hips …
```

Recovery to complete areas

It may happen that a generated HiPS contains some defects that you want to correct without restarting the whole calculation. For example, when some original images have been forgotten and then added later, it is often convenient to restart the HiPS generation, only on the concerned areas. The "region=..." parameter allows you to indicate the numbers of the HEALPix rhombs you wish to recalculate in the following form : orderN/npix1 npix2 … orderM/npix…[30]. Alternatively, you can specify the name of a MOC file describing the region to be reprocessed (FITS binary MOC format).

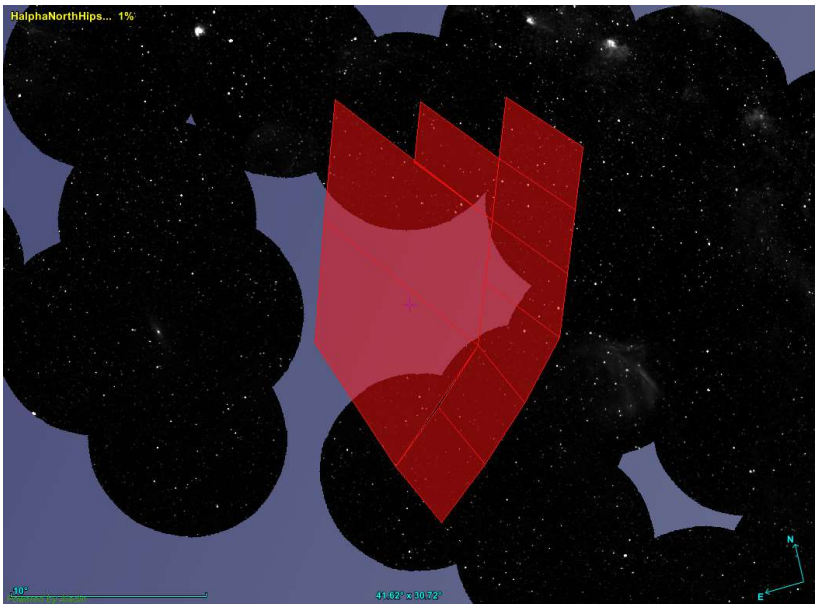
The "mode=..." parameter controls how the new tiles will be inserted into the existing HiPS: mergeKeep - keeps the pixels of the already calculated tiles and adds only the new pixels, mergeOverwrite -

---

[27] https://heasarc.gsfc.nasa.gov/docs/software/fitsio/fpack/
[28] https://arxiv.org/pdf/1112.2671
[29] This option replaces the previous "-f" option (still supported).
[30] This is the syntax of an ASCII MOC

overwrites the pixels of the already calculated tiles with the new values (this is the default), mergeMean - averages the old pixels with the new ones, mergeAdd - adds the old pixels with the new ones.

```
in="…" out="…" mode=mergeKeep "region=3/213 215 4/849 851 857 859 881 883-884"
```



Tip: *Getting the region can be facilitated via Aladin Desktop by clipping the area concerned with the "Draw" tool, then generating a MOC (Multi-Order Coverage) from the clipped area (menu Coverage -> Gererate a spatial MOC based on the selected drawing objects ».*

Pixel weighting: A replay operation will not have the same result if you still have all the original images (plus possibly some new ones), or on the contrary you only have the new images. In the first case, the final result will be identical to a complete recalculation. On the other hand, if you only have the new images, the weighting associated with each pixel will have been lost. For example, let's suppose that a HiPS pixel was obtained by averaging N original images, and that the second processing adds a new image to the area, in the first case, the final pixel following the mergeMean method will have the value *(N x oldValue) + newValue) / (N+1)* whereas in the second case it will be *(oldValue + newValue) / 2.*

In order to be able to add new images and use a weighting linked to the number of progenitors, it is necessary to set the parameter "incremental=true"[31] during the generation of the first HiPS. This option tells Hipsgen that for each FITS tile generated, it must also keep a tile of the "weights" associated with each pixel. These weight tiles will be used later in the event of a rework. Note that this method is expensive in terms of disk space as it will globally double the size of the final HiPS. If you think you don't need these weight tiles anymore, you can delete them via the "CLEANWEIGHT" action.

```
in="/data/img " out="/data/hips" incremental=true …
```

Note that Hipsgen does not take into account any additional FITS HDUs dedicated to the weights or masking associated with each original pixel. Technically the calculation is not a problem, unfortunately there are too many differences in the way these weights and masks are integrated into the FITS format. You will have to generate a set of images yourself that takes these weights and masks into account.

[31] Option "-live" in previous versions of Hipsgen (still recognised).

HiPS update

Updating a HiPS is usually done to add new images without having to completely restart the calculation[32]. Usually you no longer have the original images and you just want to add new ones. The safest method is to create a new HiPS with the new images, and then concatenate it to the original HiPS.

Concatenation of 2 HiPS

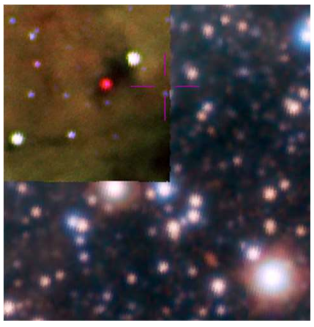
Hipsgen allows to concatenate 2 HiPS thanks to the "CONCAT" action. These 2 HiPS must necessarily be compatible, i.e. have the same maximum HiPS order and the same BITPIX.

The concatenation is done by integrating the first HiPS into the second. At the end of the operation the second HiPS will have been modified and will contain the common data. For obvious performance reasons, it is preferable to choose the smaller HiPS to be integrated into the larger one. By default the tiles of both HiPS will be merged using an average. If the "incremental=true" parameter was used when creating the two HiPS (see previous section), this average will be weighted by the number of original images concerned (progenitors). The "mode=*xxx*" parameter allows this default behaviour to be modified: mergeOverwrite: the pixels of the HiPS to be included will replace the pixels of the second HiPS even if they existed, mergeKeep: only the new pixels will be included, those already calculated will be kept as is, mergeAdd: the pixels will be added together (without weighting).

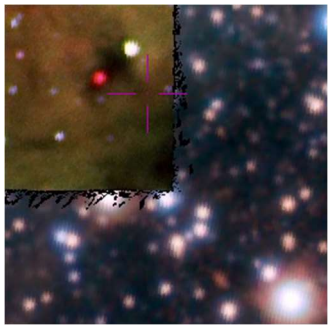
```
E.g.: in=/data/hips1 out=/data/hips2 mode=mergeOverwrite CONCAT
```

The concatenation will also integrate the original HiPS image information into the spatial index (HpxFinder) of the receiving HiPS.

Concatenation of colour HiPS: The concatenation of colour HiPS is performed in the same way. The results are much better in PNG format. Since JPG colour HiPS do not have a transparency channel, the merging will only be done for the "non-black" pixels. However, JPEG compression produces slight alterations in values, which will result in "almost" black pixels that cannot be correctly merged.



PNG HiPS



JPEG HiPS

Update by Generation & Concatenation

To make your life easier, the creation of a HiPS followed by its concatenation can be done in a single "APPEND" action. This has the advantage of ensuring that the two HiPS are compatible, as it uses the same processing parameters of the final HiPS to generate the HiPS to be merged. Of course, it is possible to modify this default behaviour, for example by using the "mode" parameter. Note that the

[32] However, it is not possible to "delete" pixels related to images that you want to discard.

presence of the original images of the receiving HiPS are not necessary for this operation, only the images of the HiPS to be created and then merged.

```
E.g.: in=/data/img out=/data/hips APPEND
```

HiPS generation by arithmetic combinations

Hipsgen offers 4 modes for combining HiPS by addition, subtraction, multiplication and division: mergeAdd, mergeSub, mergeMul, mergeDiv. These options associated with the possibility of concatenating HiPS will allow you to generate a HiPS by successive steps. An illustrative example is to obtain a HiPS from the "images" giving the counts of photons, and the exposure times.

```
1) Generation of HiPS by the sum of counts:
   in=/data/countImg out=/data/hips mode=add

2) HiPS exposure time generation:
   in=/data/timeImg out=/data/hipsTime mode=add

3) Generation of the final flux HiPS:
   in=/data/hipsTime out=/data/hips mode=div CONCAT
```

Generating a HiPS using a cluster of machines

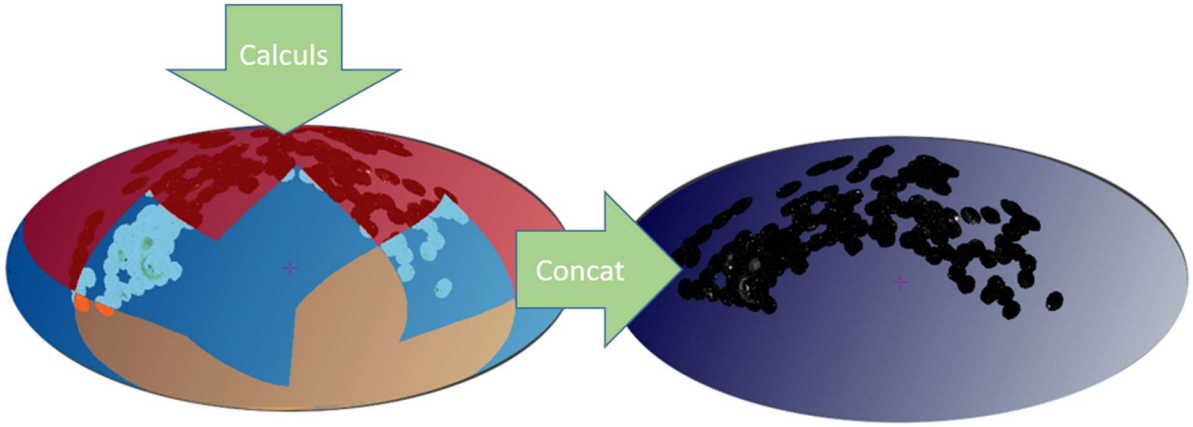
If the original data is very large, it may be appropriate to use several computing machines. Each machine will be responsible for generating a partial HiPS. These partial HiPS will be merged at the end of the processing to obtain a single final HiPS.

The strategy for distributing the work will depend on the disk capacities of each computing machine. One disk for everyone, or independent disk spaces.

Shared disc: The most practical - but perhaps not the fastest - situation is when all computing machines access the same shared disk medium. In this case, the division of the calculation will be determined by disjoint regions by using the "region" parameter. Once the calculations are complete, the partial HiPSs are merged into the final HiPS through "CONCAT" actions for each partial HiPS. If the regions have been properly divided, the method of merging has no impact on the final result. If this was not the case, it is recommended to choose "mode=overwrite" in order to correctly merge tiles that have been partially calculated several times.

```
Machine1: in=/data/img out=/data/hips  region="0/0-3"
Machine2: in=/data/img out=/data/hips2 region="0/4-8"
Machine3: in=/data/img out=/data/hips3 region="0/9-11"

Concat1: in=/data/hips2 out=/data/hips CONCAT
Concat2: in=/data/hips3 out=/data/hips CONCAT
```



Independent discs: In the case where the machines in the cluster do not share the same disk, or in order to obtain better performance on disk accesses, the division of the calculation will require either the complete duplication of the original data on each disk of each machine, or the partitioning of these original images. In the first case, the method is finally similar to that used in the case of a shared disk. In the second case, the partitioning of the original images must absolutely guarantee that all the original images concerned by one of the calculation regions are indeed on the disk of the machine in charge of the calculation.

```
Machine1: in=/data/imglot1 out=/data/hips    region="0/0-3"
Machine2: in=/data2/imglot2 out=/data2/hips2 region="0/4-8"
Machine3: in=/data3/imglot3 out=/data3/hips3 region="0/9-11"

Manual copy1 : scp machine2:/data2/hips2 machine1:/data
Manual copy2 : scp machine3:/data3/hips3 machine1:/data

Finally, on the machine1:
Concat1: in=/data/hips2 out=/data/hips CONCAT
Concat2: in=/data/hips3 out=/data/hips CONCAT
```

Tip: *It is possible to obtain the lists of images concerned by a calculation zone by diverting the use of the "INDEX" action in the following way. On a machine with all the original images, run the following command:* ***in="/data/img" out="/data/hips" order=0 INDEX****, then retrieve the names of the files concerned for each region from the contents of the spatial index tiles stored in the directory /data/hips/HpxFinder/Norder0/Dir0…*

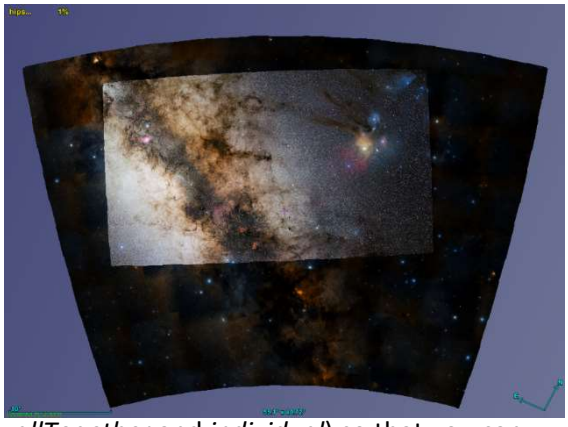
The interest of performing a HiPS calculation on several machines rather than on a single one must be evaluated on a case by case basis. The time required to select and then set up the original images, plus the time required to move the intermediate results, is not always worth the time required to calculate on a single machine. On the other hand, as a sky survey is generally available in several wavelengths, it is often preferable to parallelize the generation of HiPS by using one machine per "wavelength band" because the source data are by construction independent, and there is no merging to be done at the end of the processing.

Generation of a HiPS composed of pointed outreach images

You have a set of nice colour images (png or jpg) pointed to some astronomical objects and you want to generate a HiPS of them (e.g. the JWST images).

Prerequisites: Your images must have an astrometric calibration. See the section "Generating a HiPS from colour images".

Data preparation: One of the characteristics of outreach images is that they often focus on the same astronomical objects several times. By default, Hipsgen averages all available images. In case of astrometric shifts, and/or very heterogeneous images (wavelength, resolution), the result will be surprising, even disappointing. For these images, it is sometimes preferable to add them, one by one, to a first generated HiPS, by "overwriting" the pixels concerned using the "mode=overwrite" parameter[33]. The recommended format for such manipulation is PNG (see section "concatenation of 2 HiPS). The order in which these additional images are processed is also important. It is judicious to start with the images with the largest area on the sky and end with the smallest. Thus, you should store your images in at least two separate directories (e.g. *allTogether* and *individual*) so that you can easily manipulate them, first globally for the global generation of a HiPS, and then individually for the later additions



Hipsgen in action: The APPEND action allows you to add new images to an already generated HiPS. In the case presented here, these images will be added one by one, sequentially[34].

1) Generating a colour HiPS from a whole set of images:
```
in=/data/allTogether out=/data/hips color=png
```

2) Adding individual images by overlaying on the first HiPS:
```
in=/data/individual/image1.jpg out=/data/hips mode=overwrite APPEND
in=/data/individual/image2.jpg out=/data/hips mode=overwrite APPEND
…
```

Generation of one colour HiPS from 3 greyscale HiPS

The Hipsgen "RGB" action generates a colour HiPS from 3 previously calculated greyscale HiPS. The command is the following:

```
E.g.: inRed=hips1 inGreen=hips2 inBlue=hips3 out=hipsRGB RGB
```

Note: If the original HiPS are from a HiPS cube (see section "Generating a HiPS cube") the paths of the hips may be suffixed with "[nn]" where nn is the number of the relevant slice in the cube (starting at 0 for the first).

Basic method: This is the default method. Each colour component (red, green and blue) of the pixels to be calculated will take the pixel value of each HiPS in greyscale, scaled to 256 values. By default Hipsgen takes into account the range of values and the transfer function used to generate the compressed tiles of the source HiPS (see the pixelCut parameter). This choice can be changed using

[33] Note that the "incremental=true" parameter is useless because it has no effect (no weighting). It would only slow down the processing and increase the volume of the HiPS.
[34] It is strongly discouraged to perform APPEND operations in parallel (risk of access conflicts).

the parameters "cmRed, cmGreen and cmBlue" whose values follow the following syntax: "min max [fct]"[35] - min is the smallest pixel value to be taken into account, max the largest and fct a transfer function among: log, sqrt, linear (default), asinh, pow2. In the case where the colour HiPS to be generated uses only 2 greyscale HiPS, the third colour component will be obtained by averaging the other two.

```
E.g.: inRed=hips1 cmRed="10 1000 log" inBlue=hips3 cmBlue="40 800 log" …
```

Lupton Method: Hipsgen also offers the alternative Lupton[36] method. It uses 2 parameters for each of the colour components: luptonM (minimum) and luptonS (scale). These parameters follow the syntax "*valRed/valGreen/valBlue*" and a global coefficient « luptonQ=x » (quality). The Lupton method implements an algorithm that increases the differences between the red, green and blue components. The "minimum" determines the beginning of the range of pixels involved, the "stretch" acts on the range of pixels involved, and "Q" governs the sensitivity of the algorithm.

```
E.g.: … luptonM="10/30/45" inBlue=hips3 luptonS="1.1/1.3/1.2" luptonQ=20 …
```

Tip: *Setting these parameters is quite tricky. You will find a precious help by using Aladin Desktop (menu: Tools -> Generate a HiPS based on -> An image collection -> Generate RGB). You will be able to see immediately the result that will be obtained.*

Metadata of a HiPS

Each Hips is associated with a "properties" file that describes the HiPS: identifying it, knowing its origin, describing the related rights, etc. This information is essential if you wish to distribute your HiPS, or even make it public through the International Virtual Alliance's "HiPS network". This "properties" file contains both technical data (HiPS maximum order, tile format, etc.) automatically filled in by Hipsgen, but also description metadata that you have to update manually. The list of fields in the properties file are described in the IVOA HiPS standard. Some of them can be directly specified when launching Hipsgen: id - the identifier of the HiPS, creator - the person and/or institute that created the HiPS, and status - the status of your HiPS (private|public clonable|clonableOnce|unclonable)[37], target - the coordinates and size of a default field of view for the HiPS display, title – the HiPS title (label).

```
E.g.:in="…" out="…" id=CDS/P/myhips1 creator="Dupont [CDS]"
```

Other metadata must be edited manually in the properties file:

```
hips_copyright      = Copyright mention of the HiPS
obs_collection      = Dataset collection name
obs_description     = Dataset text description
obs_ack             = Acknowledgement mention
prov_progenitor     = Provenance of the original data (free text)
bib_reference       = Bibcode for bibliographic reference
obs_copyright       = Copyright mention of the original data
t_min               = Start time in MJD ( =(Unixtime/86400)+40587
t_max               = Stop time in MJD
```

[35] Note that you can also add an intermediate value between the "min" and the "max" to use exactly the same method that Aladin Desktop uses for its display (see the Aladin Desktop user manual).

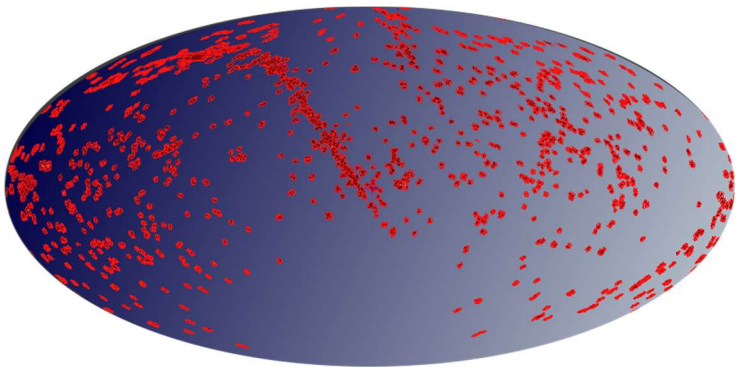
[36] https://ui.adsabs.harvard.edu/abs/2004PASP..116..133L

[37] The status private will prevent the publication of your HiPS in the context of adding it to the HiPS network IVOA; clonableOnce indicates that you allow copying and redistribution of your HiPS, but this copy cannot itself be copied. The default is "public clonableOnce".

```
obs_regime          = Waveband keyword (Radio Infrared Optical UV X-ray Gamma-ray)
em_min              = Start in spectral coordinates in meters
em_max              = Stop in spectral coordinates in meters
                        ( =2.998E8/freq in Hz, or =1.2398841929E-12*energy in MeV )
```

Moc - the spatial coverage of a HiPS

The HiPS standard recommends the addition of a "Moc.fits" file describing the spatial coverage of the HiPS. This file follows the IVOA MOC standard[38] and allows HiPS clients to know which sky areas are covered by HiPS without having to check whether tiles exist or not.



Hipsgen can explicitly generate or regenerate the "Moc.fits" file by means of the "MOC" action. In the current version of Hipsgen, this action is always executed systematically at the end of the tile generation step ("TILES" action), so there is no need to launch it specifically unless you want to modify the default generation parameters.

Thus the "mocOrder=nn" parameter allows you to impose the order of the Moc to be generated to eventually provide more detailed coverage than Hipsgen will have done by default. This parameter determines the order of the Moc to reduce its generation time. The smaller the Hips, the more accurate the Moc resolution will be. However, for large HiPS, the order can be reduced to that of the HiPS itself (but never below).

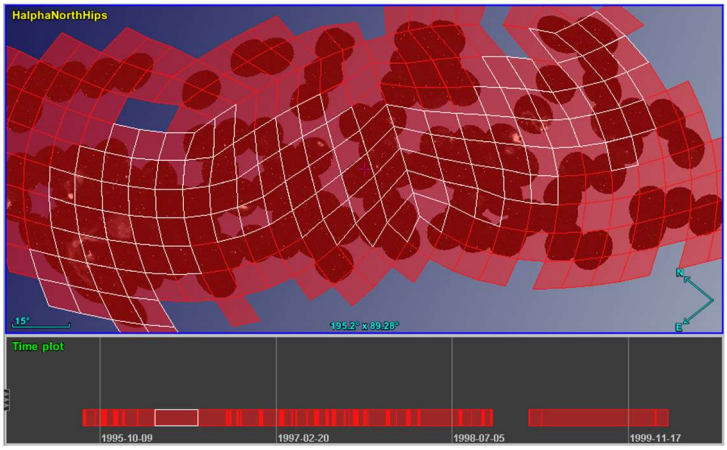
```
E.g.: out=/data/hips mocOrder=12 MOC
```

Note that calculating Moc with the same order as HiPS is much faster because Hipsgen does not need to explore the contents of each tile but simply checks that it exists.

Non-equatorial HiPS: The IVOA HiPS standards recommend that a celestial MOC must be equatorial. If the HiPS is not itself equatorial (see frame parameter) - and does not cover the whole sky - this will cause a problem of coherence and will not allow clients to know the indices of HiPS tiles. To resolve this, the "MOC" action will generate a "Moc.fits" file in the same reference system as the HiPS, and limited to the order of the HiPS (e.g.: *HiPS galactic, order 9 => Moc.fits galactic, order 9*). And in addition, it will produce a "**S**Moc.fits" file giving the spatial coverage in the equatorial system and if necessary at a better resolution than the HiPS order.

[38] https://ivoa.net/documents/MOC/

STMoc - space-time coverage?



Hipsgen is also able to generate a "STMoc.fits" file providing the coverage, not only spatial, but spatio-temporal of the original images. The "STMOC" action will take the spatial index created at the very first step of the HiPS generation (INDEX), and will use the observation date and the exposure time of each image to generate this spatio-temporal coverage. Obviously, if these characteristics are missing, or have not been recognised during the indexing stage, or not specified via the "fitsKeys = *xxx*" parameter, the STMOC generation will not be carried out. Note that the spatial and temporal resolution (mocOrder) of the generated STMOC is fixed and cannot be modified.

```
E.g.: out=/data/hips STMOC
```

This "STMoc.fits" file is not part of the IVOA HiPS 1.0 standard, but some HiPS clients are already able to use it.

Rq: Hipsgen version 12.118 and later will also update the "tmin" and "tmax" fields in the "properties" file with the minimum and maximum time bounds of the generated STMOC.

Allsky or not?

The HiPS standard offers the possibility to add an "Allsky" file when generating a HiPS. This file is a low resolution mosaic of the 768 third order tiles, assembled edge to edge. This file - in fact, these "Allsky.ext" files because associated with each tile format - are located in the "Norder3" directory. They are intended for HiPS visualisation clients that cannot display tiles of orders 0, 1 and 2 without introducing prohibitive distortions. As these clients are becoming rare, this option is less and less necessary.

Hipsgen can explicitly generate or regenerate "Allsky" files by means of the "ALLSKY" action. In the current version of Hipsgen, this action is always executed at the end of the tile generation step ("TILES" action), so there is no need to launch it specifically.

```
E.g.: out=/data/hips ALLSKY
```

Cover page: index.html

The HiPS standard requires a cover page in the form of a web file "index.html". Hipsgen generates this page automatically at the end of the tile production stage. Hipsgen provides the HTML code for viewing the HiPS using the Aladin lite web client. This page can be viewed locally, or remotely via a web server. Depending on the version of Hipsgen, the rendering of this page has evolved. Do not hesitate to perform an "UPDATE" action on your HiPS to benefit from the latest version offered by Hipsgen.

Note that you can edit and modify this page to match your needs, especially if the default layout proposed by Hipsgen does not suit you. After that, Hipsgen will inhibit any further automatic changes in order to keep your custom version. If you change your mind you will need to delete your "index.html" file before it can be re-generated (e.g. via an "UPDATE" action).

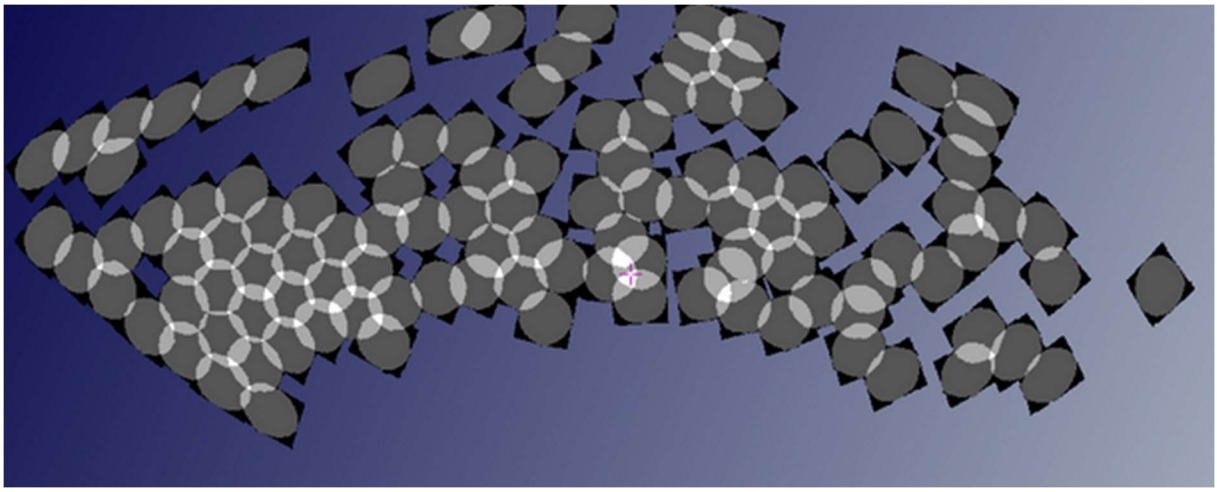
Generating a weight HiPS

The "COUNT" action (Hipsgen version 12.134 and later) adds the possibility of generating a weight HiPS, in addition to the regular HiPS. Each pixel of each tile indicates not the value of the final pixel, but the number of original images that contributed to its calculation.

If the Hipsgen settings require a weighting on the pixel calculation, typically a weighted average through the "exptime =" parameter, or a "fade-in" overlay mode through the "mode=overlayFading" parameter, the weight HiPS will provide the weight factor associated to each HiPS pixel.

Such a HiPS can be used to individually know the characteristics of the pixels in the final HiPS. It can be useful for selecting/reducing the set of original images required, as well as determining the best Hipsgen setting.

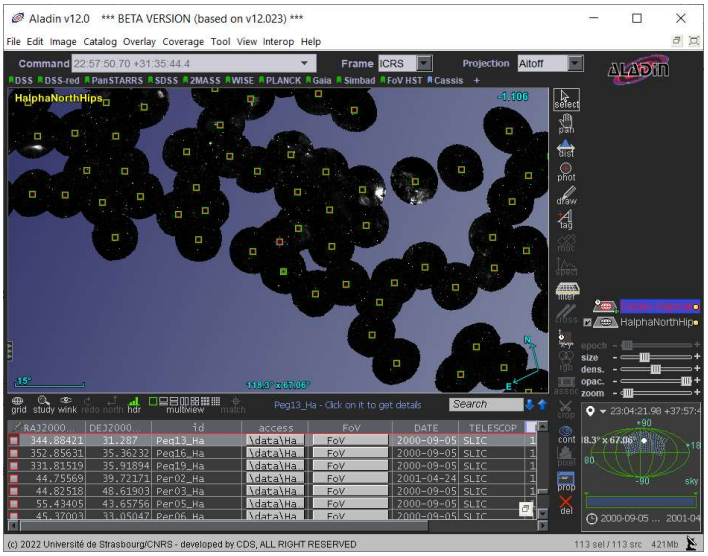
This "weight HiPS" is generated in a dedicated "HpxCounter" subdirectory.



To delete such a HiPS - typically after generation of the final HiPS - use the "CLEANCOUNT" action.

Information and links to the Progenitors of a HiPS

If desired, Hipsgen can extend the spatial indexing information created in the very first step (INDEX) to provide a convenient way to access descriptive information about the original images (called "progenitors") and direct links to them. Hipsgen will generate in the HpxFinder directory of the spatial index a HiPS tree of tiles containing not pixels but the required information. Visualisation clients compatible with this mechanism, such as Aladin Desktop, will then be able to offer their users a way to directly access the information and links of the original images used to generate the HiPS.



The descriptive information of the original images is by default the name of the image, the field of view as well as all the descriptions that Hipsgen will find in the FITS headers: the telescope, the date of the exposure, the exposure time, etc. For this, Hipsgen uses a list of keywords usually used for this information. However, if you wish to modify this list, you will have to fill in the parameter "fitsKeys=*key1 key2 ...*" in the very first INDEX step. If you have not done so at that time, you can obviously restart this action afterwards.

```
E.g.: in=/data/img out="…" fitsKeys="DATE TELESCOP" INDEX
```

FITS keywords detected by default: DATE, MJD_OBS, UTC, LST, DATE-OBS, MJD-OBS, MJD-END, OBS-DATE, DATE-END, DATEOBS1, DATEOBS2, MIDOBS, ORDATE, TIMESYS, MJDREF, JD, EXPTIME, TEXPTIME, OBSTIME, TIME-OBS, WAVELMIN, WAVELMAX, WAVELEN, TELESCOP, TELNAME

HpxFinder spatial index tile format: The format used for the spatial index tiles is JSON compatible, with the additional constraint of one record per row. It provides in this format a set of "*key = value*" pairs for each feature of the original images that intersect the spatial tile.

Example of HpxFinder tiles


```
{ "name": "And09_Ha", "path": "\data\HalphaNorth\And09_Ha.fits", "ra": "14.941473011561637",
"dec": "40.25323764636017", "cellmem": "592900", "stc": "POLYGON J2000 21.244649286615324
34.981714101299595 8.737279657225253 34.91732519744614 7.6327614938394825 45.11716085475206
22.16170676427507 45.199693481863804", "DATE": "2000-09-05", "TELESCOP": "SLIC", "DATE-OBS":
"1996-11-03T03:31:00", "EXPTIME": "360.0" }
{ "name": "And12_Ha", "path": "\data\HalphaNorth\And12_Ha.fits", "ra": "7.634980856435132",
"dec": "36.17319535985604", "cellmem": "592900", "stc": "POLYGON J2000 13.64854679555983
30.92460475628456 1.7069870688210023 30.863603081811345 0.8016234536391366 41.07006051893838
14.389415836239602 41.14458120239245", "DATE": "2000-09-05", "TELESCOP": "SLIC", "DATE-OBS":
"1996-11-04T04:30:00", "EXPTIME": "360.0" }
```


HpxFinder spatial index previously "packed": Starting with version 12.642 of Hipsgen, the HpxFinder spatial index will be packaged by default in order to store all index tiles in a few binary files (PACKINDEX action). If this is the case, the "DETAILS" action will automatically perform the reverse operation ('UNPACKINDEX' action) beforehand. See the section "*Packaging tiles*".

To format the spatial index it is necessary to execute the "DETAILS" action.

```
E.g.: in="…" out=/data/hips DETAILS
```

This action not only generates the necessary HpxFinder tree, but also sets up a "metadata.xml" file in the HpxFinder directory. This file uses a substitution mechanism and macros that will allow visualization clients to convert the content of spatial indexing tiles into a table in VOTable format (IVOA standard)[39].

Example of a "metadata.xml" file

The header of the "metadata.xml" file describes the columns that should appear in the HiPS client when the user wishes to view the characteristics of a particular original image (see Aladin Desktop illustration on previous page). This header is compliant with the IVOA VOTable document, and can use the options of this standard to generate links to the original image, either by means of a VOTable "LINK" (example below line 15), or by fields provided by the IVOA Obscore / DATALINK standard[40].

```
1 <VOTABLE>
2 <RESOURCE>
3   <TABLE>
4     <FIELD name="RA" datatype="double" precision="5" unit="deg">
5       <DESCRIPTION>Right ascension</DESCRIPTION>
6     </FIELD>
7     <FIELD name="DE" datatype="double" precision="5" unit="deg">
8       <DESCRIPTION>Declination</DESCRIPTION>
9     </FIELD>
10    <FIELD name="id" datatype="char" arraysize="13*">
11      <DESCRIPTION>Dataset name</DESCRIPTION>
12     </FIELD>
13    <FIELD name="access" datatype="char" arraysize="9*">
14      <DESCRIPTION>Display original image</DESCRIPTION>
15       <LINK content-type="image/fits" href="${access}"/>
16    </FIELD>
17    <FIELD name="FoV" datatype="char" arraysize="12*">
18       <DESCRIPTION>Field of View</DESCRIPTION>
19    </FIELD>
```

The data section of the "metadata.xml" file is reduced to a single VOTable record. This record will allow to extract the desired information from the HpxFinder tiles according to the user's consultations. To do this, each field of this single record contains a "macro" that refers to the tile values. For example the macro "$[name]" will be replaced by the value associated with the field "name" in the HpxFinder tiles (see example of HpxFinder tile on the previous page).

```
20    <DATA>
21     <TABLEDATA>
22       <TR>
23       <TD>$[ra]</TD>
24       <TD>$[dec]</TD>
25       <TD>$[name]</TD>
26       <TD>$[path:([^\[]*).*]</TD>
27       <TD>$[stc]</TD>
28       </TR>
29     </TABLEDATA>
30    </DATA>
31   </TABLE>
32 </RESOURCE>
33 </VOTABLE>
```

[39] https://ivoa.net/documents/MOC/
[40] https://www.ivoa.net/documents/DataLink

Syntax of the "metadata.xml" macros

The macros in the "metadata.xml" file use a sophisticated substitution mechanism, the basic operation of which was illustrated in the previous paragraph. Where the simple full substitution is not appropriate, a more complex syntax can be deployed.

First of all, it is possible to use several macros for the same field and to add free texts which will always appear identical. Only the macros will be substituted by the corresponding values from the HpxFinder tiles.

```
Text1 $[macro1] text2 … $[macro2]…
```

In addition, if for a given macro, the expected substitution should not take the whole value, but only a portion of it, it is possible to designate this portion by means of a suffix using the syntax of Unix regular expressions with "capture groups".

```
$[macro:regu(lar)expression]
```

If, for example, the HpxFinder tiles contain a field indicating the date of the observation (e.g. *{"DATE-OBS" : "1996-11-04T04:30:00"}*) and you want to see in the HiPS client only the year as a constant prefix (e.g. *Year 1996*) you will need to use the following macro which will extract from the "*DATE-OBS*" field all the digits up to the first hyphen, preceded by the prefix "*Year*":

```
Year $[DATE-OBS:^([0-9]+)-.*]
```

The header of the `metadata.xml` file will declare an additional field[41] :

```
20      <FIELD name="Info"/>
```

And the data section of the `metadata.xml` file will provide the additional field:

```
29          <TD>Year $[DATE-OBS:^([0-9]+)-.*]</TD>
```

Thus, depending on the designated progenitor, the HiPS client will obtain, for example, the following result:

RA	DE	id	access	FoV	Info
345.73718	37.6812	And03_Ha	\data\Ha..	FoV	Year 1996

Implementation: The tuning of the `metadata.xml' file is greatly facilitated by applying the following method: 1 - First check that the VOTable syntax is correct. To do this, either use a VOTable validator, or load the `metadata.xml` file directly into Aladin Desktop to ensure that it can read it; 2 - Use the simplest possible regular expressions, making them more complex afterwards, until you get the desired string; 3 - Don't hesitate to create "hidden" VOTable columns ("type=hidden" attribute in the VOTable FIELD) and to use these columns to build the appropriate LINK or DATALINK.

Generating a HiPS from a single image covering the full sky

Some missions provide their observations in the form of a map in Cartesian representation covering the whole sky. It is often provided in jpeg format without any associated calibration. With the "-hhhcar" option and the indication of the single image concerned in the "in" parameter, Hipsgen will

[41] The name of the column (here "Info") is independent of the name of the macros.

automatically generate the ".hhh" calibration file associated with the image, and will launch the HiPS generation process.

```
-hhhcar in="/data/skymission.jpg" out="/data/hips" …
```

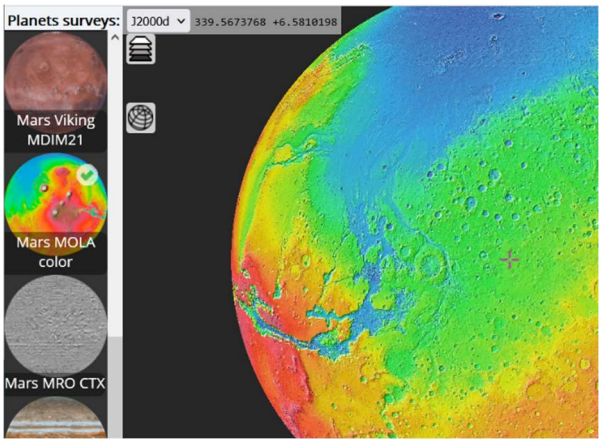
Non-equatorial image: If the image uses a non-equatorial spatial reference system, it will be necessary to use the parameter frame=*xxx* where *xxx* can take the values "galactic" or "ecliptic". Note that this parameter will also constrain the reference system of the final HiPS. Thus, if you wish to generate an equatorial HiPS from a non-equatorial image, you must proceed in two steps. First, create the "*hhh*" file in the reference system of the source image (adding the -n option will speed up this step by inhibiting the effective actions), then, with a second Hipsgen command, generate the equatorial HiPS.

```
Etape 1 : -n -hhhcar in=/data/skymission.jpg out=/data/hips frame=galactic
Etape 2 : in=/data/skymission.jpg out=/data/hips …
```

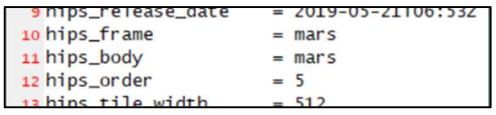
Huge image: In the case of a very large image and on a machine with limited memory resources, it will probably be necessary to proceed manually, by first cutting the original image into more easily manipulated portions. The associated "*hhh*" files will have to be adapted according to.

Generating a planetary HiPS

Originally developed for sky surveys, Hipsgen can also be used to generate planetary HiPS as long as the body concerned is reasonably spherical. This extension of HiPS usage is compliant with the IVOA HiPS 1.0 standard even though it is not specifically described. The use of Hipsgen is similar to its celestial use, but requires some minor adaptations described below. Also the HiPS clients need to be slightly adapted (see Aladin Lite planetary example on the left).



Planetary surface images generally use specific formats, e.g. PDS [42], which are not recognised by Hipsgen. Therefore, it is necessary to convert these images into a compatible format: FITS, JPEG or PNG. This step can be very complex or very simple depending on the nature of the original data. The simplest case is a Cartesian image covering the entire planetary surface. Then the method described in the previous paragraph: "*Generating a HiPS from a single image covering the whole sky*" can be applied immediately. Once the HiPS is generated, it will be necessary to manually edit the "properties" file to designate the planet concerned via a new parameter "hips_body = *xxx*" and to replace the reference system "hips_frame = equatorial" by the reference system of the planet. While waiting for an update of the IVOA HiPS standardisation document to take planets into account, it is recommended to use the name of the body (in English, and in lower case) both for the designation of the planet/body and for its spatial reference system (***sun***, ***mercury***, ***venus***, ***earth***, *moon*, ***mars***, ***ceres***, ***saturn***, *titan, dione, enceladus, iapetus, mimas, rhea, tethys,* ***jupiter***, *callisto, europa, ganymede, io,* ***uranus***, *ariel, miranda, oberon, titania, umbriel,* ***neptun***, *triton,* ***pluto***, *charon…*)



[42] https://pds.nasa.gov

Generating a HiPS from a "HEALPix map"

Some observing missions, such as PLANCK, provide sky surveys in the form of a large FITS file containing pixel values directly in HEALPix projection. This is called a "HEALPix map". Hipsgen can take such a map as input instead of conventional images. The "MAPTILES" action is dedicated to this processing. the spatial localisation step ("INDEX") is not required, and the choice of coordinate system is constrained by the one used in the HEALPix. As with a classic HiPS, it is possible to generate a set of compressed tiles via the "PNG" or "JPEG" actions.

```
Ex: in=/data/map.fits out=/data/hips MAPTILES PNG
```

It is important to note that in such a case, Hipsgen strictly preserves the values of the original map, without any resampling or interpolation.

Generating a HEALPix map from a HiPS

Hipsgen's "MAP" action generates a HEALPix map from a HiPS. A HEALPix map is a single FITS file that explicitly contains each HEALPix pixel value. It is a very similar format to HiPS except that it is limited by the size of the file (and the RAM required to manipulate it). The resolution of the HEALPix map, determined by the NSIDE, is by default 1024. The Hipsgen parameter "mapNside=*xxx*" allows you to specify a specific NSIDE value. To be identical to the original HiPS, this resolution should correspond to the formula: *mapNside = tileWidth x 2^order*.

```
in="/data/hips" out="/data/map.fits" mapNside=2048 MAP
```

This operation is only interesting if the HEALPix map produced remains of a reasonable size. An *nside* greater than 4096 is in practice not very usable.

Generating a HiPS from the deepest level tiles

The construction of a HiPS can be partially done with dedicated code (python, java, Rust...). Hipsgen will only be used to finish the job. So, if you are able to generate all the deepest level tiles of a HiPS by storing them as expected according to the HiPS IVOA standard in a *Norder**x*** directory and in the subdirectories designated *DirNN000* according to the *NpixNNNNN.fits* tile nomenclature (respectively .png or .jpg), you will be able to rebuild the full HiPS using the Hipsen "TREE" action. This action will (re)create the whole missing hierarchy, and then the additional files required.

Note that by default, the generated HiPS will use the galactic reference frame[43]. You will need to specifically specify another referential via the parameter « frame=xxx ».

```
E.g.: out=/data/hips frame=equatorial TREE
```

[43] Historical reason related to the genesis of HiPS before IVOA standardisation.

You can also use this method to apply an external filter on all the deepest order tiles. Then regenerate the hierarchy of lower order tiles by means of this "TREE" action. In this case you can force Hipsgen not to build the HiPS tree by specifying the same value as the deepest order in the "minOrder" parameter. Then, after having made the modifications to the tiles, restart Hipsgen with a "minOrder=0" this time[44].

```
Step1: in=/data/img out=/data/hips minOrder=7 order=7 INDEX TILES
Step2: treatment of the tiles of order 7
Step3: out=/data/hips minOrder=0 TREE
```

Generation of a HiPS cube from several HiPS

Hipsgen can be used to generate a HiPS cube from several HiPS already generated. For such an operation, it is essential that all your HiPS are compatible. They must share the same encodings (bitpix, bscale, bzero), the same order, the same formats and tile sizes.

To perform such an operation, you must use the "CUBE" action, indicating the list of HiPS as parameter "in". By default, the HiPS tiles will be duplicated. However, by adding the "mode=link"[45] parameter you can ask Hipsgen to create "relative symbolic links" to the original HiPS tiles. This method is much faster, but it forces you to keep the original HiPS tiles. The identifier of a HiPS cube conventionally uses the hierarchy *"…/C/…"* instead of *"…/P/…"*. (e.g. *id=CDS/C/mycube*).

```
E.g.: in=/data/hips1;…;/data/hipsN out=/data/hipscube id=CDS/C/mycube CUBE
```

Generating a HiPS cube from FITS cubes

Hipsgen can also be used to generate a HiPS cube directly from a set of FITS cubes. The generation is done exactly the same way as for a classical pixel HiPS, with the same actions and parameters. As mentioned in the previous paragraph, the identifier of a HiPS cube conventionally uses the hierarchy *"…/C/…"*.

Prerequisites: All FITS cubes must share the same depth (same number of planes) [46].

```
E.g.: in=/data/fitscubes out=/data/hipscube id=CDS/C/mycube INDEX TILES …
```

When the cubes are pointed ("spread out") and not a mosaic of cubes, the proportion of "sparse" tiles[47] can become significant as they are multiplied by the depth of the cube. The use of a compression method for FITS tiles should be considered if you wish to keep them (see section "Compression of FITS tiles").

Note that HiPS clients able to display HiPS cubes are still rare. Unlike Aladin Desktop, a non-compatible client will only display the first plane of the HiPS cube.

---

[44] This trick will save you 10% of the generation time. However, if you do not intend to modify the deepest order tiles, this 2-step process should be avoided as it would increase the generation time by 6%.

[45] In some versions of the Windows OS, the generation of symbolic links is only allowed if you have administrative rights.

[46] The HiPS cubes defined by the HiPS 1.0 standard only consider the number of the plane, not its physical correspondence (wavelength, time, ...). The cube mosaic does not perform resampling in the 3rd dimension.

[47] A "sparse" tile contains a high proportion of meaningless pixels ("blank" value).

Note: Starting with version 12.5, Hipsgen implements a prototype version of HiPS dedicated to frequency cubes: “HiPS3D”. This extension is expected to eventually complement or even replace the “classic” HiPS cubes (see the “HiPS3D” section at the end of this document).

Duplication of a HiPS

Duplicating a HiPS can be done by any dedicated copy tool, e.g. "*cp -R /dir .*" for a local copy or across the network via "*rsync -Hav host:/dir .*". If the HiPS has been installed on an http server, it is also possible to use a "*wget*" command without the need to log in on the remote machine and with the required parameters to browse and copy the entire tree.

For partial copying, e.g. by focusing only on certain tile formats, or by discarding the deepest orders, the standard tools can become complex to parameterize. Hipsgen's "MIRROR" action allows such partial or full copy operations to be performed[48]. For this operation, the “in” parameter will fill in the base URL of the HiPS available on the http server, or, in the case of a local duplication, the HiPS directory. The "format=*xxx*"[49] parameter will allow you to indicate the list of asked tile formats (fits, png and/or jpeg). The "order=*nn*" parameter will allow you to limit the depth of the replicated HiPS orders. It is also possible to take into account only one area of the sky by entering the "region=..." parameter (see section « HiPS computation recovery » above).

At the end of the copy operation, Hipsgen will update the properties file to adjust the parameters affected by the copy (hips_order, hips_format, hips_status) and verify the conformity of the check codes to ensure that the copy is correct.

```
E.g.1: in="/data/hips" out="/data/myhips" format=png MIRROR
E.g.2: in="http://server/hips" out="/data/myhips" order=4 MIRROR
```

Recovery of a duplication: If the duplication of a remote copy is interrupted, Hipsgen MIRROR can be restarted to complete the copy. By default, Hipsgen only checks that the size of the already copied tiles is reasonable, and if necessary restarts the copy of these tiles. The "fastCheck=false" parameter can be set to additionally check that the date and size is consistent with the remote tile. However, this test slows down the copy process considerably[50]. Note that in the case of a full copy, the HiPS numerical code is validated at the end of the copy, which will detect a possible erroneous copy.

Copying spread over several directories: Duplicating a HiPS may require a lot of disk space. If the target directory is not big enough, it is possible to ask Hipsgen to split the copy over several target directories. Thus the parameter "mirrorSplit=size;altpath ..." allows you to indicate one or more alternative directories needed for the copy if the default directory does not have the required size. For example *mirrorSplit="10g;/data/hips-ext1 200g;/data/hips-ext2"* will cause the first 10GB to be copied into the default directory "*out*", then 200GB into the alternative directory */data/hips-ext1*, and all the rest into */data/hips-ext2*.

```
E.g.:  in=http://server/hips  out=/data/myhips  mirrorSplit="100g;/data1/althips"
       MIRROR
```

[48] Hipsgen runs multiple copy threads in parallel and adjusts the number of threads according to the instantaneous performance of the network. This technique works well on both fast and slow networks.
[49] Previously called “mirrorFormat” for Hipsgen versions prior to 12.135
[50] Checking the size and date on the remote server often takes as long as the copy itself.

HiPS compliance check

A HiPS consists of a large number of files which must comply with the standard set by the International Virtual Observatory Alliance.

Compliance with the standard: A first level of verification of a HiPS will be to validate its compliance with the IVOA standard. The Hipsgen LINT action checks each required element of the standard document and produces a compliance report for each item checked.

```
E.g.: out=/data/hips LINT
```

If your HiPS was generated before the current IVOA HiPS standard, Hipsgen's "UPDATE" action will allow you to update it to comply with the current standard. This operation will also set up - if missing - the digital keys (checkcodes and DATASUM) that recent versions of Hipsgen use to check the conformity of a HiPS (see following paragraphs).

```
E.g.: out=/data/hips UPDATE
```

In addition, following a copy or a move, it may be useful to be able to check the conformity of the copy. Two complementary techniques are possible, one very fast, the other slower but more complete.

Quick check: The first method checks that all files are present, and have kept their original size. The "CHECKCODE" action, executed by default by Hipsgen at the end of the generation of a HiPS, will add the keyword "hips_check_code" in the properties file by associating a numerical key specific to each set of tiles (e.g.: "*png:3137881707 fits:3716611556*")[51]. These numerical keys are correlated to the number and size of all the tiles in each format. Therefore, after a copy, the execution of the "CHECK" action will recalculate these keys according to the tiles present and check whether the result corresponds to the keys recorded in the "properties" file. If this is not the case, the copy does not conform. This method is very fast, but it is not totally sure in the sense that a file could have been altered without changing its size. This case is very unlikely in the case of a copy or transfer, but it is possible in other cases of alteration (involuntary permutations of FITS files with the same size, ransomware, ...)

```
E.g.: out=/data/hips CHECK
```

Full check: The second method is more reliable because it will calculate a numerical key directly on the content of the tiles, and not simply on their size. When generating the tiles and following the FITS standard, Hipsgen generates a DATASUM key specific to the content of each FITS tile, and stores it in its header. Thus, after a copy, the execution of the "CHECKDATASUM" action will recalculate each DATASUM of each FITS tile and check its correspondence with what was previously memorized. If this is not the case, the copy of the FITS tiles is not correct, even if the number and size of the tiles are correct. This method is very robust[52] but unfortunately a hundred times slower than the previous method. Additionally, it only takes into account the FITS.

```
E.g.: out=/data/hips CHECKDATASUM
```

---

[51] The CHECKCODE action will also update the properties file with the total number of tiles (hips_nb_tiles) and the total size of the HiPS (hips_estsize) provided in KB.

[52] The DATASUM method, is very reliable, but not absolutely guaranteed. For example, it cannot detect a permutation of values (2 pixels of 4 bytes inverted). But such a probability is very low (unless you do it on purpose).

In the case of HiPS generated with earlier versions of Hipsgen, or by other tools not using a FITS digital key, the "UPDATEDATASUM" action will allow you to calculate and store the DATASUMs without having to regenerate the FITS tiles. This takes approximately 25% longer than the check itself (and approximately half the time it would have taken to regenerate all the FITS.

Random check: In the case of "ransomware" viruses, Hipsgen also allows to "scan" a HiPS by means of the CHECKFAST action to quickly check if it has not been altered or encrypted. The method consists of randomly checking the consistency of certain elements of the HiPS in order to avoid making a backup of a HiPS that has been corrupted. If there is a problem, Hipsgen displays the following message "ERROR: HiPS is corrupted".

```
E.g.: out="/data/hips" CHECKFAST
```

HiPS tile packaging (Hipsgen version 12.135 and later)

The HiPS IVOA standard describes the methods for accessing HiPS elements (tiles, metadata, coverage, etc.) via an HTTP API. Consequently, it may seem natural to physically store these elements as individual files in the server's file system. This simplicity of storage is one of the strengths of HiPS technology. However, it is also possible to use other means of storing HiPS elements. This could be a database providing the requested elements, or specific software dedicated to the on-the-fly generation of tiles and other requested elements.

Thus, in order to provide an alternative to storing individual tiles and considerably reduce the number of physical files needed for storage, once the HiPS has been generated, the "PACK" action will replace all the tiles in each "DirNNNN" sub-directory with a single binary file containing the aggregation of the tiles concerned, for each tile format available.

```
E.g.: out=/data/hips PACK
```

If necessary, the "UNPACK" action will restore the original distribution[53].

```
E.g.: out=/data/hips UNPACK
```

These two actions can be controlled by the "format=..." parameter to specify which tile types are concerned. The value of this parameter will take the form of a series of words separated by spaces: *fits* - FITS tiles, *png* - PNG tiles, *jpeg* - JPEG tiles. By default, all represented formats will be taken into account.

```
E.g.: out=/data/hips format="jpeg fits" PACK
```

Runtime and recovery: The "PACK" and "UNPACK" actions require the complete reading and writing of all the tiles concerned. This is a fairly lengthy operation, although Hipsgen will start the process by estimating the most suitable number of threads. If the process is interrupted prematurely, it is always possible to restart the action, or cancel the work already carried out using the complementary action.

HpxFinder spatial index tile packaging: Packaging operations can also be applied to spatial index tiles (see "INDEX" action) using the 'PACKINDEX' and "UNPACKINDEX" actions. Unlike image tile packaging,

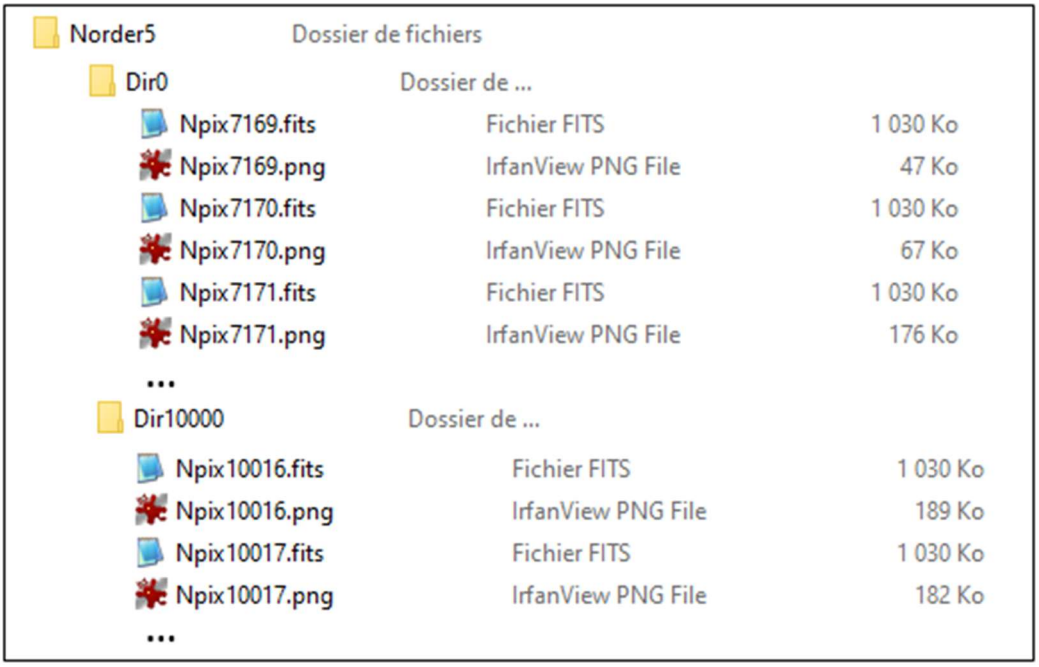
[53] A "packaged" HiPS cannot be updated directly.

this operation is fast and replaces a large number of small files with a few binary files. It is now applied by default when using Hipsgen (version > 12.642) without specifying any specific actions.

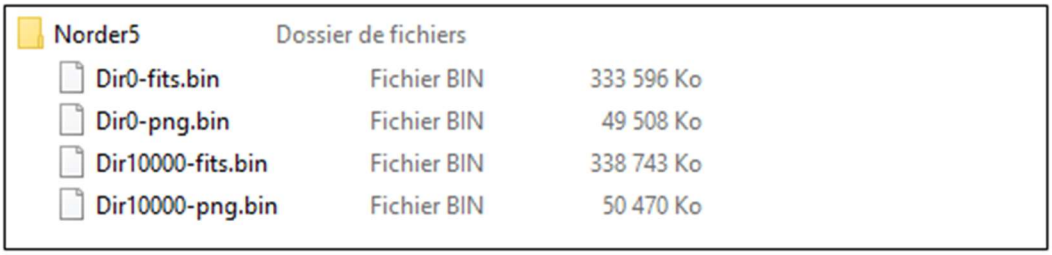
Binary file names: The binary files generated by "PACK" are prefixed with the directory name and suffixed with "-ext.bin" where "*ext*" corresponds to the format (*fits*, *png*, *jpg*).

```
E.g.: Dir10000-png.bin
```

Example: DirNNNN directory structure before the "PACK" action



The same HiPS after the "PACK" action :



Binary file format: The binary files potentially contain 10,000 tiles, those found in the DirNNNNN directory for a given extension. These files consist of two parts: "the tiles" and "the index". The first part of the file stores the tiles, as is, one after the other in ascending numerical order of their indices. The index memorizes the positions of the stored tiles and ends with a specific magic code. All integer encodings are in little endian.

```
  0 ..  W : 1st tile stored in binary
W+1 ..  X : 2nd tile   "      "   "
X+1 ..    : 3rd tile   "      "   "
            …

  Y .. Y+7 : End address of 1st tile used +1 (=W+1)
Y+8  .. Y+15: End address of 2nd tile (=X+1)
```

```
                …
      L-8 .. L-7 : Index of 1st used item in index (between 0 and 9999)
      L-6 .. L-5 : Index of last item used (between 0 and 9999)
      L-4 .. L-1 : Magic code (= PCK2)
```

Tile extraction: The distribution of a " packed " HiPS requires the use of a small CGI (or equivalent) so that the http server can extract the requested tiles from the binary files[54]. This program must implement the following reading algorithm:

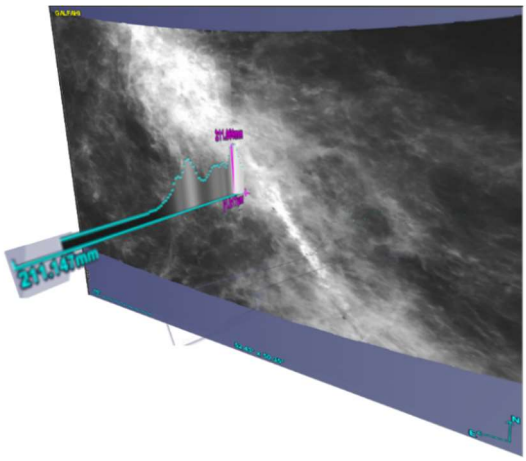
Extracting the tile with index "npix" and extension "ext" involves three read actions in the file DirNNNNN-ext.bin [55]

1) Read the last 8 bytes of the file (size L) to find out the index of the first (first) and last (last) tiles stored, and deduce the position of the start of the index (Y).
    a. Optional Magic Code check: the last 4 bytes must be "PCK2".
    b. Y = L – 8 – (last-first+1)*8
2) Reads the start and end positions of the required tile from the index:
    a. start : at the address: Y + (npix%10000-first-1)*8 (*)
    b. end : on the following 8 bytes
3) Extraction of the tile will then be possible by knowing its position in the binary file (start) as well as its size (end-start) (which may be zero if the tile is not there)

(*) if npix%10000 is less than first-1 or greater than last, then the tile is not there, and if npix%10000 is equal to first-1, the tile starts at the beginning of the file.

"HiPS 3D" (Hipsgen 12.5 and later)

A " HiPS 3D " is a prototype generalization of HiPS for cubic data, where the 3rd axis is physically calibrated. It requires a Hipsgen version greater than or equal to 12.5. A HiPS3D uses the resampling and hierarchization principles of a standard HiPS, but applies them both spatially and to the third axis. At present (July 2025), only the frequency dimension is supported. The temporal dimension will come later.



Demonstration: The document available at https://aladin.cds.unistra.fr/java/TutoHiPS3Den.pdf gives a concrete and fairly comprehensive overview of the potential that HiPS3Ds offer.

Prototype code: HiPS3D" extensions are still under development. Therefore, you need a prototype version of Hipsgen as indicated in the tutorial above. At present, only the AladinProto.jar code available at https://aladin.cds.unistra.fr/java/AladinProto.jar is capable of generating and displaying these new HiPS. It should also be noted that HiPS3D is not (yet?) standardized by the IVOA.

[54] It will take about twenty milliseconds longer to extract and publish a tile than with traditional file access.
[55] NNNNN = (npix%10000) x 10000 (integer modulo).

How to use: HiPS3D is generated from a collection of cubes with a frequency calibration recognized by Hipsgen (CTYPE3 in VELO, VRAD, WAV or FREQ). It simply requires the dedicated "-hips3d" option.

```
E.g.: -hips3d in=/data/img out=/data/hips INDEX TILES …
```

Dedicated parameters: Several new parameters have been added to modify Hipsgen's default behavior when generating a HiPS3D:

- tileDepth=*nn* : Like all HiPS, a HiPS3D is made up of tiles, but these tiles now have a thickness. By default, Hipsgen uses tiles with a thickness of 32 and a size of 256 by 256, i.e. 2Mpixels. You can adjust these values using the existing "tileWidth=*nn*" parameter, combined with this new "tileDepth=*nn*" parameter. These two values must necessarily be powers of 2.
- orderFreq=*nn*: this parameter adjusts the maximum frequency resolution taken into account by Hipsgen. It has the same function as "order=*nn*", but for the frequency dimension. To adjust this parameter, the easiest way is to run the INDEX action once and note the automatically determined frequency order and the impact this will have on the size of the final HiPS[56]. Then, if necessary, modify this default value by a few orders to oversample or undersample your cube survey. Note that decreasing by one frequency order reduces the size (and generation time) of the generated HiSP3D by a factor of 2.
- restFreq=*freq*: when generating a HiPS3D from velocity cubes, Hipsgen needs to know the rest frequency of the chemical line under study (e.g. HI = 1.420405752E9 Hz). This is usually indicated in the FITS header, but unfortunately under various possible keywords[57]. If the authors of your data have used a keyword unknown to Hipsgen, or if no indication is given, you can still explicitly mention the rest frequency value via the "restFreq=*freq*" parameter, where *freq* is indicated in hertz. This parameter will only be taken into account by default. It will not replace an existing value.
- -notrim: to save time and disk space, HiPS3D tiles are "trimmed" by default (see section "*Control and performance - Edge removal*"). The "-notrim" option inhibits this behavior in order to generate FITS tiles whose storage size on disk corresponds to the expected number of pixels (voxels), regardless of the presence or absence of edges.

Usage restrictions: Some Hipsgen actions are not compatible with HiPS3D, notably MAP, PACK, CUBE, RGB, LINT, COUNT. An error message is generated when attempting to.

Hipsgen "traces"

Hipsgen is a very verbose tool and provides a lot of information during processing. An understanding of the major steps in the process is a real plus for using this tool efficiently.

Launch

```
INFO  : Starting HipsGen 20/12/22 19:46:21 (based on Aladin v12.023)...
OPTION: in=HalphaNorth
OPTION: out=hips
INFO  : Action => INDEX: Build spatial index (in HpxFinder directory) + MOC index
INFO  : Action => TILES: Build all true value pixel tiles + Allsky + MOC
INFO  : Action => PNG: Build all preview tiles (PNG) + Allsky.png
INFO  : Action => CHECKCODE: Compute+store the check codes (and the size) associated to the target HiPS
INFO  : Action => DETAILS: Adapt HiPS index for supporting the "detail table" facility
```

[56] Example: "INFO : HiPS frequential **order 17** (freq res orig:48.8aHz hips:32.8aHz **x1.49**)"
[57] Hipsgen 12.630 and later take these keywords into account: RESTFREQ, RESTFRQ, OBSFREQ et RESTWAV

Spatial Indexing - INDEX

```
RUN   : ============================================== INDEX  ==============================================
INFO  : Partitioning large original image files in blocks of 4096x4096 pixels
INFO  : Output directory: .\hips
INFO  : Pre-existing HpxFinder index => will add new images only...
INFO  : Use this reference image => HalphaNorth\And03_Ha.fits
INFO  : Max Order=3 => Pixel angular resolution=51.53"
INFO  : Tile Order=9 => tile size: 512x512 pixels
INFO  : MEF strategy => extension 0, otherwise 1
INFO  : Extended metadata extraction based on FITS keys: DATE, TELESCOP, DATE-OBS, EXPTIME
INFO  : HiPS coordinate frame => equatorial
.
STAT  : 26 files in 893ms => 3,67Mpix using 14,85MB => biggest: [385x385 x4]
...
INFO  : Max original image overlay estimation (4096x4096 pixel blocks from original images): 16 => may required 1
GB per thread
INFO  : MOC Index done in 15ms: mocOrder=3  frame=C size=624B
STAT  : 113 files in 4s 751ms => 23.8/s => 15,97Mpix using 64,56MB => biggest: [385x385 x4]
DONE  : INDEX done (in 4s 753ms)
```

Generation of FITS – TILES

```
RUN   : ============================================== TILES  ==============================================
INFO  : Output directory: .\hips
INFO  : Reference image: HalphaNorth\And03_Ha.fits
INFO  : Set default target => 345.73718 +37.6812
INFO  : BITPIX found in the reference image => -32
INFO  : Data range [-601.8 .. 1783], pixel cut [-4.691 .. 402.8]
INFO  : Overlay mode (progenitors): OVERLAY_MEAN: Mean of the progenitor pixel values
INFO  : Merge mode (tiles): MERGE_OVERWRITE: Replace existing pixel values if the new value is not BLANK
INFO  : Hierarchy mode (tree): TREE_MEAN: The mean of the 4 sublevel pixel values
INFO  : Frame (HiPS coordinate reference frame) => equatorial
INFO  : Creating 257 .fits tiles (order=3)...
INFO  : BITPIX = -32 (no conversion)
INFO  : BLANK=NaN
INFO  : Building tiles thanks to 8 threads
INFO  : Available RAM: 7,89GB => Cache size: 300 items / 5,26GB
.........................
STAT  : 160+10/257 tiles + 77 nodes in 26s 536ms (66.1%) by 8/8 threads
STAT  : Cache: 190 items/300 using 108,11MB/5,26GB freeRAM=7,38GB (opened=190 reused=806 released=0)
....................
STAT  : Tile overlay stats : max overlays=16, 254 in one step, 0 in multi steps
STAT  : Tiles trim method saves 0KB
STAT  : Pixel times: Original images=15,97Mpix => 333,81Kpix/s  Low tiles=61,25Mpix (x3.834) => 1,25Mpix/s
INFO  : MOC generation based on fits tiles (deep resolution mocOrder=7)...
...
INFO  : MOC done in 2s 174ms: mocOrder=7 size=25,38KB
INFO  : Creating Allsky...
INFO  : properties & index.html files created or updated
DONE  : ALLSKY file done
DONE  : TILES done (in 52s 402ms)
```

PNG tile generation

```
RUN   : ================================================ PNG  ================================================
INFO  : Output directory: .\hips
INFO  : Hierarchy mode (tree): TREE_MEAN: The mean of the 4 sublevel pixel values
INFO  : Map pixel cut [-4.691 .. 402.8] to [1..255] (linear)
INFO  : Building tiles thanks to 8 threads
.....
STAT  : 245/257 tiles in 2s 423ms (95.3%)  endsIn:118ms
INFO  : properties & index.html files created or updated
STAT  : 245/257 tiles in 2s 423ms (95.3%)  endsIn:118ms
DONE  : PNG done (in 4s 839ms)
```

Generation of check codes - CHECKCODE

```
RUN   : ============================================ CHECKCODE  ============================================
INFO  : Output directory: .\hips
INFO  : Scanning png tiles...
INFO  : 356 png files for 28,68MB
INFO  : Check code for png tiles: 3137881707
INFO  : Scanning fits tiles...
INFO  : 356 fits files for 369,1MB
INFO  : Check code for fits tiles: 3716611556
INFO  : Full HiPS size: 397,78MB
INFO  : Check codes: png:3137881707 fits:3716611556
INFO  : Check codes and HiPS metrics stored/updated in properties file
STAT  : 712 files scanned
DONE  : CHECKCODE done (in 80ms)
```

Generation of link tiles to progenitors – DETAILS

```
RUN   : ============================================== DETAILS  ==============================================
INFO  : Output directory: .\hips
INFO  : Order retrieved from HpxFinder => 3
INFO  : Detail table min order determined by "minOrder" parameter => 0
INFO  : Mapping hpxFinder/metadata.xml file has been generated
STAT  : 257/257 tiles in 117ms (100.0%)  endsIn:
DONE  : DETAILS done (in 127ms)
INFO  : Tip: Edit the "properties" file for describing your HiPS (full description, copyright, ...)
DONE  : ================================== THE END (done in 1m 2s)  ==================================
```

The properties file in detail

The "properties" file contains all the HiPS metadata: the origin of the images, technical elements related to the generation of the HiPS, its description, the history of the processing carried out as well as information intended for clients. Some fields have a controlled vocabulary to be consulted in the HiPS IVOA standardisation document, others are free. Here is an example of a fairly complete properties file, with comments...

Description and origin: The fields grouped below give the HiPS identifier, its title (one line), the data collection from which it came, a fairly complete description (one small paragraph), the rights holder, a suggested credit, the wavelength, a bibliographic reference, and finally the data type.

```
creator_did          = ivo://CDS/P/2MASS/K
obs_title            = 2MASS K (2.16um)
obs_collection       = The Two Micron All Sky Survey - K band (2MASS K)
obs_description      = 2MASS has uniformly scanned the entire sky in three ...
obs_copyright        = University of Massachusetts & IPAC/Caltech
obs_copyright_url    = http://www.ipac.caltech.edu/2mass/
obs_ack              = IPAC/NASA
prov_progenitor      = Caltech
bib_reference        = 2006AJ....131.1163S
bib_reference_url    = http://simbad.u-strasbg.fr/simbad/sim-ref?bibcode=2006A
dataproduct_type     = image
```

Original coverage and resolutions: The following lines group the coverage fields: wavelength, time range and sky area. The next two fields give the angular size of an original pixel and its coding mode.

```
obs_regime           = Infrared
em_min               = 2.015E-6
em_max               = 2.303E-6
t_min                = 50600
t_max                = 51941
moc_sky_fraction     = 1
s_pixel_scale        = 2.777E-4
data_pixel_bitpix    = -32
```

HiPS origin and rights: Fields beginning with 'hips_' refer to the HiPS itself, notably the creation information: tools, standard reference number, dates, persons, rights holder.

```
hips_builder         = Aladin/HipsGen v11.023
hips_version         = 1.4
hips_creation_date   = 2013-01-14T09:45Z
hips_release_date    = 2021-02-23T01:29Z
hips_creator         = CDS (Oberto A.)
hips_copyright       = CNRS/Unistra
hips_status          = public master clonableOnce
```

Technical fields of HiPS: The technical elements are given below. These technical elements are described in detail in this manual.

```
27 hips_pixel_bitpix    = -32
28 hips_frame           = equatorial
29 hips_order           = 9
30 hips_order_min       = 0
31 hips_tile_width      = 512
32 hips_tile_format     = jpeg fits
```

Generation mode fields: The following fields provide information on the methods used to generate the HiPS.

```
33 hips_sampling        = bilinear
34 hips_overlay         = overlayMean mergeOverwriteTile treeMean
35 hips_skyval_method   = SKYVAL
36 hips_skyval_value    = -0.5 100.0 -2660.728 9046.069
37 hips_pixel_cut       = -0.5 40
38 hips_data_range      = -2661 9046
```

Measurement fields: The following fields provide the HiPS pixel angular size, the total HiPS volume (in KB), the number of tiles, the numerical validation codes and a suggested default view field.

```
39 hips_pixel_scale     = 2.236E-4
40 hips_estsize         = 4706425658
41 hips_nb_tiles        = 315200120
42 hips_check_code      = fits:3568836873 jpg:4537812981
43 hips_initial_fov     = 58.63230142835039
44 hips_initial_ra      = 266.40499479
45 hips_initial_dec     = -28.936173970
```

Processing history: Each time Hipsgen is used on this HiPS, the date and parameters of the command line - including the parameter file - are logged to provide a processing history.

```
46 hipsgen_date         = 2020-08-04T10:56Z
47 hipsgen_params       = cache=CACHE-TODEL-K cacheRemoveOnExit=true in=2MASSk
48 hipsgen_date_1       = 2020-10-04T11:03Z
49 hipsgen_param_1      = out=Hips-K3 creator_did=ivo://CDS-test/P/2MASS/K UPD
```

Fields for clients: The properties file can contain fields for clients or even for a specific client. Below, the "client_category" field is for Aladin Desktop to present the HiPS in the appropriate branch of its resource tree.

```
50 client_category      = Image/Infrared/2MASS
```

Hipsgen's actions and parameters (v.12.642)

Options

```
   -clean  : Delete previous computations
   -n      : Just print process information, but do not execute it
   -color  : Colorize console log messages
   -nocolor: Uncolorize console log messages
   -nice   : [MIRROR] Slow download for avoiding to overload remote http server
   -notouch: Do not touch the hips_release_date
   -hhhcar : [INDEX] Generate hhh file for an all sky image
   -notrim : [TILES,CONCAT,APPEND] Do not 'trim' FITS tiles
   -trim   : [TILES,CONCAT,APPEND]  'trim' FITS tiles
   -gzip   : [TILES,CONCAT,APPEND] Gzip FITS tiles
   -hips3d : HiPS3D generation dedicated to Space-Frequency cubes (prototype)
   -d      : Debug messages
   -h      : Inline help
   -man    : Full inline man (may be followed by a parameter|action for a full explanation)
```

Actions

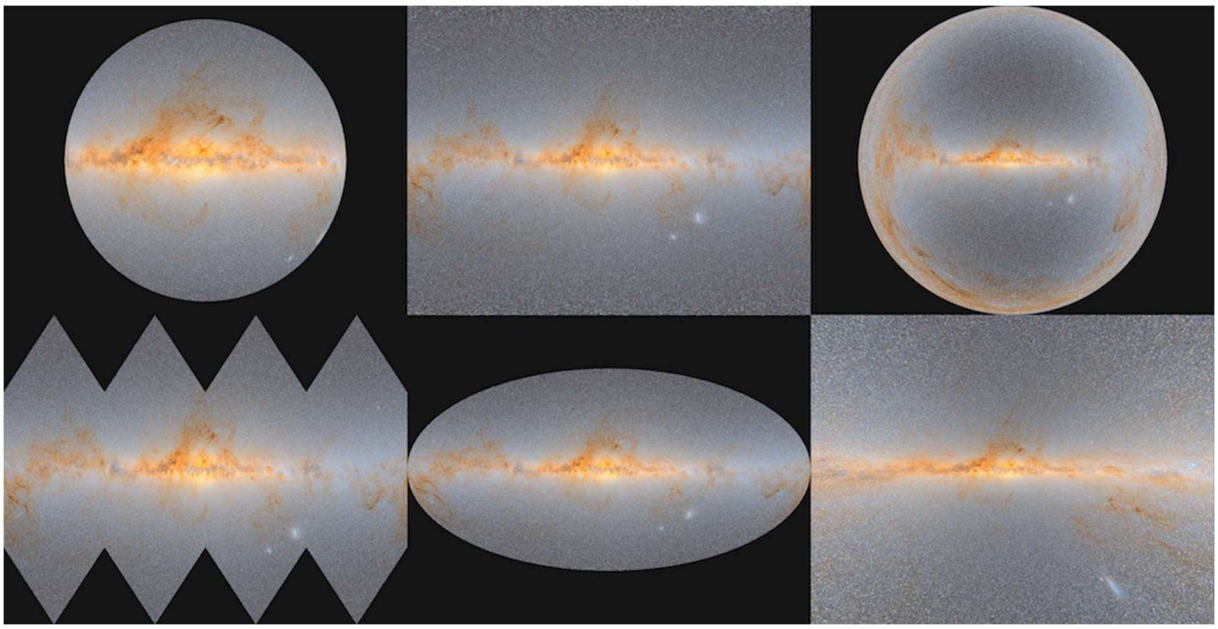
```
(by default: "INDEX TILES PNG CHECKCODE PACKINDEX "):
   INDEX        : Build spatial index
   TILES        : Build all true value pixel tiles
   PNG          : Build PNG preview tiles
   JPEG         : Build JPG preview tiles
   MOC          : Regenerate the HiPS spatial coverage (MOC)
   MAP          : Build an HEALPix map from the HiPS tiles
   ALLSKY       : (Re)build all Allsky files
   CLEAN        : Delete all HiPS files (except properties file)
   CLEANINDEX   : Delete spatial index
   CLEANFITS    : Delete all FITS tiles
   CLEANJPEG    : Delete all JPEG tiles
   CLEANPNG     : Delete all PNG tiles
   CLEANWEIGHT  : Delete all WEIGHT tiles
   TREE         : (Re)build HiPS hierarchy from existing tiles
   APPEND       : Append new images to an already existing HiPS
   CONCAT       : Concatenate two HiPS
   CUBE         : Create a HiPS cube based on a list of HiPS
   DETAILS      : Extends HiPS spatial index for supporting 'progenitor' facility
   STMOC        : Build a STMOC.fits based on HpxFinder tile descriptions
   UPDATE       : Upgrade HiPS metadata additionnal files to the last HiPS standard
   CHECKCODE    : Compute and store the check codes
   MIRROR       : Duplication of a HiPS (local or remote)
   RGB          : Build and RGB HiPS based on 2 or 3 other HiPS
   CHECK        : Basic HiPS integrity check
   CHECKDATASUM : HiPS FITS tiles full integrity check
   CHECKFAST    : Fast HiPS integrity test
   LINT         : Check HiPS IVOA 1.0 standard compatibility
   MAPTILES     : Build HiPS tiles from a HEALPix FITS map
   COUNT        : Build progenitor counting HiPS
   PACK         : HiPS packing
   PACKINDEX    : HiPS index packing
   UNPACK       : HiPS unpacking
   UNPACKINDEX  : HiPS index unpacking
```

Parameters

```
   in=dir              : Source directory
   out=dir             : Ouput directory
   order=nn            : HiPS order
   minOrder=nn         : HiPS min order
   frame=equatorial|galactic|ecliptic: HiPS coordinate frame
   tileWidth=nn        : HiPS tile width
```

```
orderFreq=nn        : HiPS frequential order
bitpix=8|16|32|64|-32|-64: HiPS bitpix
dataRange=min max   : Original pixel range
pixelCut=[min[%] max[%]] [byRegion[/size]] [fct]: 8 bits pixel mapping method
img=filename        : Reference image for default initializations
hdu=n1,n2-n3,...|all: List of FITS HDU numbers (original images)
blank=nn|key        : Alternative BLANK value (or alternate BLANK fits keyword)
restfreq=freq       : Default rest freq. (Hz) if the RESTFRQ (or equiv.) key is missing
validRange=min max  : Range of valid pixels
skyVal=key|auto|%info|%min %max: Background removal method
expTime=key         : Method of adjusting the exposure time
maxRatio=nn         : Image source pixel ratio test
fov=true|x1,y1..    : Masks on the original images.
border=nn|N W S E   : Edge removal
shape=rectangle|ellipse: Image FoV
mode=m1,m2..        : Coadd pixel modes
incremental=false|true: Incremental HiPS
region=inline moc|moc.fits: Working region
partitioning=false|nnn: Splitting large original images into blocks
maxThread=nn        : CPU thread limitation
fastCheck=true|false: Mirror check method
fitsKeys=key1,key2...: FITS keywords for image characteristics extraction
id=AUTH/P/...       : HiPS identifier
title=title         : HiPS title
creator=name        : HiPS creator
target=ra dec [radius [freq]]: Default HiPS target
status=private|public [clonable|clonableOnce|unclonable]: HiPS status
color=jpeg|png      : Tile format of a colour HiPS
inRed=hipspath      : Red HiPS path
inGreen=hipspath    : Green HiPS path
inBlue=hipspath     : Blue HiPS path
cmRed=min [mid] max [fct]: Red color mapping
cmGreen=min [mid] max [fct]: Green color mapping
cmBlue=min [mid] max [fct]: Blue color mapping
luptonQ=x           : Q coef Lupton RGB builder
luptonS=x/x/x       : Scale coefs Lupton RGB builder
luptonM=x/x/x       : M coefs Lupton RGB builder
cache=dir           : Alternative cache directory
cacheSize=nnMB      : Alternative cache size limit
cacheRemoveOnExit=true|false: Removing cache disk control
mocOrder=nn [<nnMB] : Specifical MOC order an/or size limit
mapNside=nn         : HEALPix map NSIDE
format=fmt1 fmt2 ...: Tile formats to process
mirrorSplit=size;altPath ...: Multi disk partition split
pilot=nn            : Pilot HiPS for testing
```

Table of contents



Hipsgen – User manual
January 2023 version – with additions