Euro 3D format description

2D image + cube

Yannick COPIN*

June 18, 2001

1 Introduction

I describe in this document the so-called "*2D image + cube*" format (hereafter 2Di+C format), the third of the 3 proposed formats for the Euro 3D datacube format, along with "*3D cube*" (hereafter 3Dc format) and "*2D image + table*" (hereafter 2Di+T format). I will try to describe the concept and present its pros and cons without any considerations for the technical feasability.

Hereafter, I call a "*datacube*" the structure that will contain the full 3D information $(\alpha, \delta, \lambda)$ as obtained with an integral field spectrograph. In formats 2Di+C and 2Di+T, such a datacube is dissociated in a "*spectral stage*" – a structure mainly dedicated to the storage of the spectral dimension of the datacube – and a "*spatial stage*" – a structure mainly dedicated to the storage of the spatial dimensions of the datacube. In a 3Dc format, such a distinction does not occur.

The spectral stage of the 2Di+C format – all the spectra being stored together as the lines of a 2D image ($x \equiv \lambda, y \equiv$ lens number) – is similar to the one prescribed in the 2Di+T format, so I will not elaborate on this aspect. I will rather emphasize the differences *table vs. cube* for the spatial stage of the datacube in this 2 dissociated formats.

2 Discussion

The spatial storage of the "*2D image + table*" consists of a table listing, for each of the spectrum – identified by its *lens number* `NO` – stored in the

*Sterrewacht Leiden, ycopin@strw.leidenuniv.nl

2D image, the position and other spectrum-related quantities (e.g., total flux $\mathcal{F}$, mean stellar velocity $\bar{V}$, etc.):

```
# NO     X      Y      FLUX     V       ...
  ...
  33    -0.1   -0.1    111     -1199   ...
  34    -0.1    0.0     92     -1221   ...
  35    -0.1    0.1     87     -1276   ...
  36     0.0   -0.1    123     -1234   ...
  37     0.0    0.0    112     -1256   ...
  38     0.0    0.1     99     -1278   ...
  39     0.1   -0.1    109     -1233   ...
  40     0.1    0.0     89     -1212   ...
  41     0.1    0.1     80     -1282   ...
  ...
```

This kind of storage has a *very low spatial content*: all the lens-related quantities ($\mathcal{F}$, $\bar{V}$, etc.) are described as function of the *lens number*, and not as function of sky position. Therefore, to get a quantity at a given position requires a 2-step procedure:

1. find the lens at this given position,
2. read the quantity related to this lens.

To emphasize the spatial aspect of the storage, I once proposed to store the lens-related quantities no more as function of lens number but as function of lens position: we therefore switch from a fundamentally 1D table – all quantities function of lens number – to a 2D table – all quantities function of lens position, e.g.:

```
NO   ...  -0.1   0.0   0.1  ...
 ...
-0.1 ...   33    34    35   ...
 0.0 ...   36    37    38   ...
 0.1 ...   39    40    41   ...
 ...
```

or

```
 FLUX  ...  -0.1   0.0   0.1  ...
  ...
 -0.1  ...   111    92    87  ...
  0.0  ...   123   112    99  ...
  0.1  ...   109    89    80  ...
  ...
```

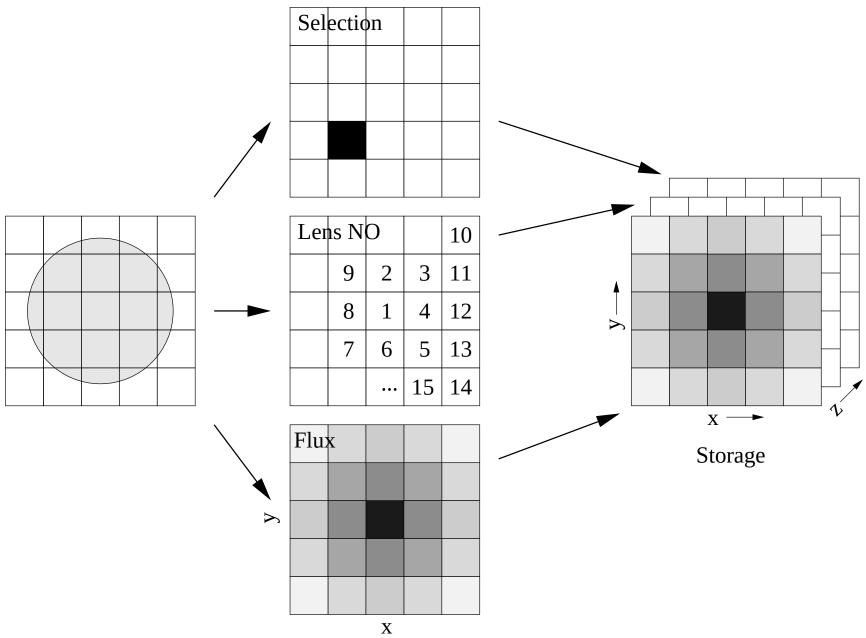

Figure 1: Construction of the 3D table: *left:* coverage of the galaxy by the lens array; *middle:* each lens gives various quantites (e.g., selection flag, lens number, total flux, etc.), stored in 2D images; *right:* the 2D images are packed in a 3D cube, which is the so-called *3D table*.

If the sampling is *regular* in both direction x and y, each of these "2D tables" forms a usual 2D image – the intensity of each pixel storing the value at its position – and these images – one for each lens-related quantities – can be packed together along the z-axis to form a *cube*.

Thus, we have replaced each 1D column in the associated table of the 2Di+T format by a 2D z-slice of the cube in the 2Di+C format, therefore giving more emphasis to the spatial directions.

3 Description

As mentionned before, the table of the 2Di+T format is replaced in the 2Di+C by a cube, which dimensions (x, y) give the sky position and which third dimension z is related to the various lens-related quantities, such as lens number, $\mathcal{F}$, $\bar{V}$, etc. (see Fig. 1).

This format requires that the spatial sampling is regular in each direction,

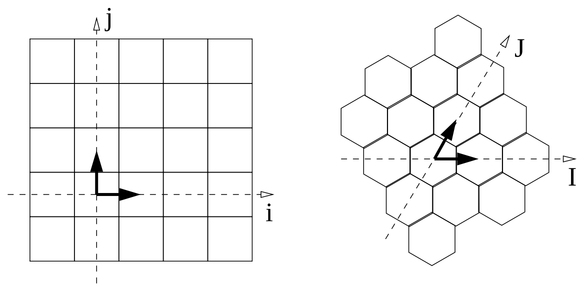

Figure 2: Coordinates used for the square tiling (*left*) and the hexagonal tiling (*right*).

but not necessary the same. There are practically two ways to tessellate the plane, with *squares* or with *hexagones*. Both tessellation can be handled by the 2Di+C cube, by using the appropriate coordinates for the 2 dimensions of the z-slices (see Fig. 2):

square packing: the cartesian coordinates (i, j),

hexagonal packing: the I, J coordinates.

A selection (good/bad flag) over the lenses to discard for further analysis, or more generally a "quality weight" associated to each lens, could be implemented by inserting a "selection mask" in the 3D table (see *top panel* of Fig. 1).

4 Pros & Cons

Here are few points *for* and *against* this format.

Pros

- This format emphasizes the spatial directions of the datacube with respect to the "lens-list approach" of the 2Di+T format;
- In the case of a square packing, each z-slice of the cube directly gives the *reconstructed image* of the considered quantity (see *bottom panel* of Fig. 1);
- Appliable to the two major tessellations of the plane.

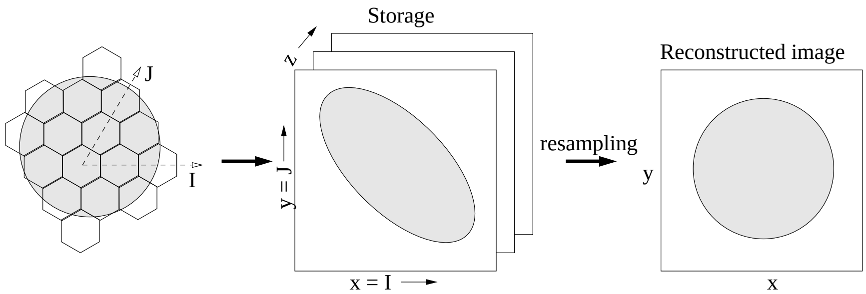

Figure 3: Distortion of the z-slice image in the case of an hexagonal packing.

Cons

- If some lenses are selected out, an interpolation is needed to produce a hole-free reconstructed image. In any case, a check over the selection mask is needed before issuing a reconstructed image: therefore, a simple z-slice extraction of the 3D table is not enough;
- In the case of an hexagonal packing, a z-slice of the cube gives a *geo;etrically distorted* image of the considered quantity (see Fig. 3). An *undistorted* reconstructed image thus requires a resampling of the z-slice;
- The format is not adapted if the plane tessellation is not regular (e.g., after spatial binning), or if the coverage of the sky is not 100 % (e.g. with an hexagonal packing of circular fibers);
- To my knowledge, there is no tool (libraries) to create and manipulate the 3D table as described here.