WPs “3.1.3.1” “5.3.1.1” & “5.3.1.2”
Data Cube Tools:

Cube Spectrum Analysis toolbox
User requirement

Version 1.2

9 April 2008

Alain Gueguen

1 Purpose

The purpose of this document is to design and specify the software for the analysis & display of the CubeDataset . The software should propose many tools for using CubeDataSet defined in the doc describing the various WP as:

1. Tool to generate integrated intensity map from CubeDataSet
2. Tool to create Position Velocity Maps
3. Tool to generate channel maps from a CubeDataSet
4. Tool to run line-fitting package on all elements of a CubeDataSet.
5. Tool to create map from fitted parameters
6. Tool to create maps based on observed line properties of individual spectra
7. Extract single spectrum from Cube data.

2 References

[RD-01] Herschel image datasets specification
Doc ID: PICC-KL-TN-015
Date: 30. May. 2005

[RD-02] Simple and Sliced Products Wim de meester 2007_07_25_WDM_Data_Format.pdf

[RD-03] HIPE presentation hyperspectral workshop ESTEC 01/30/2008

3 History

This document his updated from feed back and/or after meetings.

3.1 feed backs

- July 2007 Marc Sauvage
- July 2007 Alessandra Contursi
- End July 2007 Peter Davis

3.2 meetings & workshop

- ICC meeting Garching June 2007
- July 25&26 Marseilles 3D spectrum visualization workshop
- CSDT meeting 12-14 September oxford
- SPG meeting 17-18 September Garching
- Hyperspectral visualization and analysis – 3 Garching 24-25 October 2007
- Hyperspectral visualization and analysis – 4 ESTEC noordwijk 30 January 2008

3.3 Location of the last version of this document.

The official version of this document is in the livelink, as PDF:

The last updated version is available in the herschel wikipage in the Sandbox section of the author :

http://www.herschel.be/twiki/bin/view/Sandbox/AlainGueguenSandbox

the old version are archived at the same place.

4 General introduction

4.1 Fusion of Work Packages:

This document combines the user requirements and some development propositions for 3 Extended DP Work Packages: 3.1.3.1 (Cube Dataset analysis tools), 5.3.1.1 (PACS spectral visualisation) and 5.3.1.2 (PACS Cube Dataset visualisation). These 3 WPs take as input the Simple and Sliced Cube products and are strongly related to each other. They will share a lot of tools at the display level but also at the computation level. In addition, they will probably be used linked to each other and in a common way by the users. Therefore, it seems logical to develop them at the same time and in the same environment. depending of the final definition of the GUI for the Extended DP (integration in HIPE or not) this kind of development will help to define a common acces to these features

These work packages are not to be mixed up with the HIFI work package 3.1.1.x spectrum analysis toolbox, a generic Herschel tool for the analysis of spectra (e.g. continuum fit, line flux, etc.). However, both tool boxes might have some functions in common (e.g. line fit?), and the output of the spectral cube analysis tool certainly will have to feed into the spectrum analysis tool. In order to save development time we work in collaboration and re-use as much as possible the tool resulting of these HIFI WP.

Finally, these tools and the requirements on them have to be coordinated with the requirements on the extended DP work package 3.2.3.x (Cube Dataset visualisation), developed by PACS (S. Regibo). The various analysis tools spectral visualisation and dataset analysis should finally be accessible from a GUI for the “Cube Dataset visualisation”.

4.2 Compatibility of the Cube Spectrum analysis tools.

The PACS pipeline will finally deliver a cube for the spectrum. this will be the native (and only one) format for the input of the ***Cube Spectrum Analysis Tools.*** This toolbox will be available for the three instruments (**PACS** ,**SPIRE** and **HIFI**), it's therefore the responsibility of the 2 other instruments to deliver there data in the good “Cube” format to be used in this toolbox.

for PACS last step of the pipeline to send the data in the *SimpleCube* will be done in the “***CubeBuilder***” module.

Features the ***Cube Spectrum Analysis Tools*** will offer will result of a combination of the needs of all the users of the different instrument.

4.3 Plan for the releases

This document describe the different features the CubeSpectralAnalysistoolbox should offer. For each of this features there will be a default mode and an advanced mode.

The default mode itself could be more or less complex and need some parameters. Therefore the first release of the “CubespectralAnalysistoolbox” will offer the default mode and the various advanced mode will be available in the next releases.

5 CubeDataSet Description

The original Cubedataset product described in the document [RD-01], can be a combination of various elementary Datasets: cube of spectra, intensity map, masks. Originaly this should have been the standard input for the tools described in this document.

From this original format 2 new types of products were defined and will be used (depending of the size of the data), the SimpleCube and SlicedCube.

The cube will contains some images aligned on the sky with associated projection, errors, masks and exposure, distributed in time or in Spectral unit (Wavelength wavenumber, frequencies).

The final result of the PACS pipeline will be put in a SimpleCube (or Sliced). The first release of the Simple Cube is available since mid of october 2007 and CubeSpectralAnalysistoolbox will use it. The development started therefore from the description and is done with a strong interaction with the developers of the Cubedataset and the ones of the **Cubebuilder**, the last step of the pipeline.

Since the first developments were made with dataset, an interface for the various tools will also be able to work with "Classical" Dataset Arrays, dataset. Depending of the evolution of the PCSS, new interfaces (at the constructor level) will be added for new input format data.

In order to simplify the development, the various tools will all consider the same internal format for the data.

A 3D cube for the data (set of layers of aligned images, distributed in wavelength) associated to a 1D array for the wavelength (or the Z axis Unit) (depending of the information available on July 07). The final cube which will be delivered by the Cubebuilder (P.Appleton) will be the new internal format when available.

Remark:

> For the PACS specto-imager the wavelength for one layer could be inhomogeneous, at the "raw" level of the data. We consider here that, if existing, this in-homogeneousness was previously corrected.

The user manual and the developers notes will describe precisely the generic input format (cube and or array3D) to allow the users and developers of the other Herschel's instrument to export there data to this format. For the final Users, the *User Manual and the developers notes* Should list precisely the various format used by the different instrument, and, if implemented, the external format compatible.

6 User Requirements for the Spectrum Cube visualization tools

6.1 Introductions

6.1.1 Pacs raw data preparation

The *Simple and Sliced-cube* coming from the spectrum part of the Pacs pipeline will contain globally set of images “aligned on the sky”, distributed in spectral unit along the third axis.

The PACS spectrometer will deliver basically N “sets” of 16*25 “matrix of pixel”. This elementary signal will be converted as 5*5*(N*16) cube (*N=integer*), each of the N*16 layers corresponding to an unit wavelength (after calibration). This transformation will be made by the “CubeBuilder”. A tool will also offer the possibility to join various basic cube in order to work with bigger images, (“map” of 5*5*N pixel cube), for this we have to check that the correction of the projection and of the distortion of the field of view was correctly done in the pipeline.

For PACS we assume at the beginning that for the cubes the “Z” axis unit will be a wavelength. The “Spectral data cube” is therefore a “Level 2” data.

Finally, for the “spectral” data cube, the CubeDataSet visualization tools should offer many features listed and described above.

6.1.2 Extraction of elementary images, for other input, if needed.

Like said before the other instrument must provide there data in a cube format corrected of the instrument effects. They will be manipulated internally as a Double3d array and/or as a set of spectrum1D or spectrum2D.

In order to be compatible with other instrument than the ones from Herschel (for comparison) The “*Cube spectral analysis tool*” will probably accept in direct input 3D arrays (associated with a 1D array for the wavelength) and (Simple or Sliced).

6.1.3 Output and Results

All the tools of the *CubeSpectrumAnalysistoolbox* will delivers some “Graphical” results (Spectrum line, 2D map, mosaic cubes ....).

For all these results, we will provide

- A “save as ” feature to save
  - a HCSS Product in a fits file
- A hard copy process
- A high quality electronic format good enough to be use for publication (high precision Postscript).
- A Save Script Button, which will allow the user to redo the same process on the cube with the same parameters in a “jide” session or in a “Hipe” Session or in an environment offering JYTHON execution capabilities with the HCSS librairies.

Since the result could have various format, each of these “Save As” will be specific, The “print” feature will also depend of the result format. At this step we need to define for each features of the spectrum analysis toolbox the complete list of the needed outputs .

Some of the tools could also deliver some set of many scalar values per pixels. for this kind of tools the best format is probably a fits table or an ASCII file. For the graphical display of these tools the various component of the sets should be processed separately.

6.2 Extract a Single pixel spectrum from Cube data

6.2.1 Precondition

The extraction of a Spectrum from images in a cube will be possible only if:

- The cube contain images
  - Aligned on the sky. (first assumption at this date) (If the images are not initially aligned this should be done in a independent process before the extraction)
  - The WCS are available for all the layer (the same for all if image are aligned)
- The Z axis unit is Spectral dimension with information on the UNIT (wavelength, wavenumber, frequencies, velocities)

Like defined in the Cube specification and description.

This tool will be available for all the "Cube" data of the HCSS if they respect the previous precondition.

6.2.2 Features

The **Default** mode of this spectral extraction will be the raw spectrum:

- The intensity of the spectrum at a given wavelength will be the intensity of the chosen pixel, or sky coordinates, in the image at this wavelength.
- The spectrum will be computed on one pixel given by the user (given in X,Y or RA/DEC in the GUI).
- Output Data will be a ***Spectrum1D***, the spectral unit must be stored as **Double (**at least from the HIFI instrument and by extension for all data using frequencies) or in an better precision format.
- The initialization of the wavelength will be made directly from the Simplecube or by using a wavelength array (when using a double3d for the intensities).
- A filter could be applied on the raw spectrum just to "remove" the noise. The result of this filtered spectrum would be display in the result as a second layer in the graphical display of the spectrum (PlotXY).

As **Advanced** mode Many options could be proposed for improved raw single pixel spectrum extraction such as:

- "Smoothing" (?) the spectrum. This kind of filtering of the spectrum need a list of smoothing method. (its probably one of the requirement of the §5.5).
- "fitting" the raw spectrum. we need a list of the fitting method to applicable. (this is probably also part of the §5.5).
- Rejection of noisy layer. (If necessary) this must be done via a list of layer to reject. Due to the definition of the Cubebuilder and of the SimpleCube the non valid values would be flagged in the "FLAG" Section of the SimpleCube and some EROR values will be available in the Error section. if these block are not empty the extraction will take care of these points.

QUESTION: what find of value must be put in place of a "non valid" one ?
- null
- Zero
- mean value from previous and next ?

In the main window is a Real Time single spectrum extraction (see §7.10), therefore the current feature is more dedicated to display a spectrum in a bigger zoom and in addition to apply filters and fit on it. Therefore an interface will allow the user to choose to apply a set of filter/fits.

these operation will re-use as much as possible the toolbox developed by M.Melchior in Zurich and R.Assendrop and others at SRON.

6.2.3 Interface

- *Command line:*
  - The extraction of a single pixel spectrum from a SimpleCube is available as a Task ***"ExtractSinglePixelSpectrumTask"*** so it can be executed from a command line interface (JIDE) and the GUI of the CubeSpectrumAnalysistoolbox is basically a graphical way to fill and initialize the various parameter needed to execute this task. the GUI can also be considered as a way to select the first parameters to execute latter some "Batch" on many cube resulting of many observation from the JIDE editor.

    The parameters will be:

    - The cube itself "cube" a Double3d
    - The "spectralDimension" a String (FREQUENCY, WAVELENGTH,NONE)
    - The Spectral unit a String (Hertz, Ghz,Meter,Micrometer.....)
    - The wavelength of the layers if the dimension is "wavelength", a Double1d (empty of the dimension is frequencies)
    - The frequencies of the layers if the dimension is FREQUENCY "freq" a Double1d (empty of the dimension is wavelength)
    - A flag for the Real time use of this task "realTime" a Boolean
    - The X position of the pixel to be read "posX" an Integer
    - The Y position of the pixel to be read "posY" an Integer
    - The inside size of spectrum (or number of layer of the cube) "sizeOfSpectrum" an Integer
    - The X dimension of the cube "dimX" an Integer
    - The Y dimension of the cube "dimY" an Integer
    - The return value will be a Double1D array during the development phase, and latter depending of the final choice will become a **Spectrum1D** containing the couple intensity wavelength in the final version. (or the user could choose the kind of result he want.)
  - The Spectrum Extraction process must allow to apply some process on the extracted spectrum: the various process will be available as various task, one for the filtering one for the fitting.

    These Task will

    - Take the spectrum
      - In a Double1d
      - In a couple of Double1d (Intensities, spectral unit values)
    - Take a Set of parameter to define the properties of the input parameters such as:
      - The spectrum "Dimension"

- The spectrum Unit
- The unit of the intensities
- Convert the data in a Spectrum1d in order to call the needed fitter and filter tools
- The Filter ot fitter to be applied
  - the model
    - For the *smoothing* mode we need a list of model and parameters
    - The first filter are
      - A gaussian filter, with a width defined in “Channel” , centered.
      - A BoxCarFilter I.E a gate function defined also in channel values and centered.
  - the numeric values defining the model (width ...)
- Convert the resulting spectrum in the output format.

- **QUESTION**: What should be the return format of the extracted and processed spectrum for the various Tasks:
- At this date this will be a Double2D containing couples (I,Lamdba) or (I, Frequencies) or (I,Velocities) associated to some parameters such as:
- unit of the spectral unit
- dimension of the spectral unit
- size & position of the spectrum extracted in the cube
- properties of the filter or fitter
- on other possibility could be to put the values in a couple of Double1d

- *Display*
  - In the global display the user see the spectrum corresponding to the current position of the mouse in the image. a click on the mouse fix the spectrum at the last position
  - The graphical interface show in a text-field, the properties of the pixel under the mouse or after the click the last position extrated.
  - If a smooth or a fitt (notavailable at this date) is selected the spectrum is processed. The processed spectrum is shown is the same PlotXY in a second layer.
  - The GUI offer a “SAVE DATA” button. This button save the various spectrum and values in a FITS file. The Double 1d or 2d will be put in fits table , the parameters will be store in the fits header .

QUESTION: Here we need an information about the “astrometry” to put in the header: X,Y of the pixel Corresponding RA/DEC ..

  - A button “SAVE SCRIPT” save a jython script which can be executed directly from JIDE. the script initialize the parameters for the extraction task and if needed the processing task.
  - A button “Print” print the PLOTXY and the spectrum.
  - Since the spectrum are shown in a classical PLOTXY all the features of this plotXy are available form the right click of the mouse like the definition of the titles subttitles the names of the axis and of the layers .....

6.3 Extract an "averaged" spectrum from Cube data

This feature is an evolution of the previous one: Instead of reading the spectrum for one pixel along the Z axis, this feature allow to extract the spectrum from a region (circle or rectangle at this date).

6.3.1 Precondition

The precondition for this point are exactly the same that for the single pixel spectrum.

6.3.2 Features

The basic features of this tool are based on the ones of the single pixel spectrum, with in addition the need for a region to read. This region can be a:

- Circle :

  The user draw a "free floating" circle on a layer, with a center and a radius which can be "real" values, not only integer. The extraction consider "partial" pixels in a linear approximation (The effective area is therefore a polygon), in the future the process will consider the real area of the circle.

- Circle with a weighted function (top-hat, gauss , empirical beamshape ...).

  This feature can be considered as an evolution of the Circle extraction but as a strong physical significance . The list of function is needed.

- Rectangle:

  The user draw a rectangle which is fixed on the pixel limits.

- Polygon

  The user draw a polygon on the image, the spectrum in computed on the pixels inside the polygon

The Smoothing mode can be more complicated since it can be applied in 2 dimension:

- Along the Z axis, to smooth or filter the "spectral noise" (To Be corrected)
- On the images to smooth or filter the noise related to the images themselves (To Be Corrected and reformulated)

The extraction of the Averaged spectra re-use the Class developed for the single pixel extraction. A dedicated method will be added to construct the Spectrum1D resulting of the multiple single spectra.

All the options proposed at the previous step (single pixel) should be proposed for this averaged spectrum extraction. If one of the option is accepted and implemented it should be implemented for the 2 kind of spectrum extraction, as long as it have a physical signification.

6.3.3 Interface

- *Command line:*
  - The extraction of an "averaged on many pixel" spectrum is available as a Task : "**ExtractRegionPixelSpectrumTask**" from a command line interface so it can be executed from a command line interface (JIDE) and from the GUI "CubeSpectrumAnalysistoolbox". This GUI is basically a way to fill and initialize the various parameter needed to execute this task. like for the single pixel spectrum extraction the GUI can also be considered as a way to select the first parameters to execute latter some "Batch" on many cube resulting of many observation from the JIDE editor.

- The parameters for this task are:
  - The cube itself "cube" a Double3d
  - The "spectralDimension" a String (FREQUENCY, WAVELENGTH,NONE)
  - The Spectral unit a String (Hertz, Ghz,Meter,Micrometer.....)
  - The wavelength of the layers if the dimension is “wavelength”, a Double1d (empty of the dimension is frequencies)
  - The frequencies of the layers if the dimension is FREQUENCY "freq" a Double1d (empty of the dimension is wavelength)
  - The array of pixel to be read as (X,Y,Weight) as Double2d
  - The inside size of spectrum (or number of layer of the cube) "sizeOfSpectrum" an Integer
  - The X dimension of the cube "dimX" an Integer
  - The Y dimension of the cube "dimY" an Integer
  - The Return value will be a Double1D array during the development phase, and latter depending of the final choice can become a **Spectrum1D** containing the couple intensity wavelength in the final version. (or the user could choose the kind of result he want.)
- Since the “region” to be extracted is a list of pixels this array can define non contiguous pixels. This is a possibility of the “command line” mode which at this date is not available from the GUI
- The “averaged” Spectrum Extraction process is compatible with the filter and Fitter task described in the section 6.2. the same question are therefore applicable for this point.

- *Display*:
  - In the global display the user will click the position he want the spectrum.
  - The graphical interface will show in a text-field, the properties of the selected region , the eventually weight method and or filter fitt method. The extraction itself will be executed when clicking on the mouse. For the smoothing and filtering mode a second extraction could be launched by clicking a button.
  - The result will be display as a spectrum in a PlotXY in the GUI. It will also be possible to save, export , print the resulting spectrum in many format:
    - Print directly
    - Export for publication in postscript and other format (to be defined) in a quality compatible with the publication requirement.

6.4 Creating an integrated intensity map from CubeDataSet

6.4.1 Introduction

The integrated intensity map result of the integration of the complete spectrum or of a range of this spectrum pixel per pixel of the complete cube. the choice between complete spectrum or range of spectral units will result of some controls on the availability of the wavelength information and of the astrometry, the informations about the satellite and instrument and about the observation itself. Each instrument can have a default mode which can be modified by the users.

6.4.2 Features

6.4.2.1 Default mode

- For the PACS data:
  - The Integrated Intensity is the intensity resulting of the complete range of wavelength present in the cube. This means that this functionality, for this instrument in its mode, considers the complete CubeDataSet for all the wavelengths (or all the times).
  - The integration process will consider a constant response for all the wavelengths , so the integration will be a simple sum.
- For the other instruments:
  - The “scientific” or “usable” data can be only part of the complete range of wavelength. Since the default usable range of the different instrument will always be the same, the default mode will offer to reduce the integration range on “all”, “user specified range”, “SPIRE” or “HIFI” (To be defined).
  - At this date the integration won't use any “response” or “weight” function.

The integration process will be done in some Tasks which will be call from a GUI or from the command line.

In the GUI, The resulting spectrum will be

- Shown in a display window.

From the command line and in the GUI the result can be:

- Saved as a ,
  - HCSS product (Double2d SimpleImage), in a fits file
- Printed
- Saved it as ,
  - Postscript format
  - An image file
- Returned in the JIDE environment (or to the environment from which the GUI was launched) as a Double2D dataset (or the default image type for the HCSS) or in a SimpleImage.

Since the unit of the "Z" direction in a CubeDataSet can be wavelength, frequencies or time, (available as a component of the metadata section), there must be a control on this unit. The physical result of an integration in time is not the same as an integration in wavelength, so the result of the integration must be stored with the Unit information. (the final cube delivered by the CubeBuilder should contain only wavelength).

- QUESTIONS:
  - For the instruments requiring an integration on a "Range" of wavelength , what are the limits (compared to the complete range available contained in the cube) ?

6.4.2.2 Advanced mode

In addition of the default mode some more specialized integration mode could eventually be proposed. If there is a demand for an advanced mode we'll need with this demand some extra informations, parameters and rules like:

- Weight of the different frames (referring to the spectral sensibility for the wavelength for example) (array or function).
- **Filtering on a Signal/Noise ratio or else (If an image of the cube is to noisy it must be possible to reject it from the integration). In this case, parameters could be a list (of images to be rejected or of images to be selected).**
- Addition with a "convolution" or a smoothing.
- **Integration on a range, not on all the domain of wavelength (on a continuous range or on a "discontinuous" range (with selection in a list of available layers example)) (in the case where the Z unit is wavelength). (this is already part of the default mode)**

  In fact, part of this, correspond to an other feature, the channel map.

This list of features must be discussed and completed with the final users requirements. At this date no one of these propositions will be implemented if there is no explicit need.

- QUESTIONS:
  - For the various needs of an probable advanced mode , what are the precise requirement?
    - for the Weight function:
      - Is there an instrumental effect which could be calibrated and corrected
      - Is there a specific response functionality
    - for a "Conditional Filter" what could be the rules
    - What kind of smoothing filter ?
    - ...

6.5 Creating a Position Velocity Maps

6.5.1 Definition of a Position Velocity Map

2 kind of Position Velocity Map are identified:

6.5.1.1 Position Velocity Map along an axis

This kind of **Position Velocity Map** is a diagram of the statistic of the velocity along an axis defined by the user. (See for an example http://arxiv.org/PS_cache/astro-ph/pdf/0508/0508619.pdf

N.peretto et all 2006 page 21) "Probing the formation of intermediate- to high-mass stars in protoclusters -A detailed millimeter study of the NGC 2264 clumps".

I.E it's a map of the relative velocities compared to a reference in the image and it's computed from the Doppler effect. The result is therefore a view of the "histograms" of the spectrum, converted in speed, for all the pixels along the axis.

6.5.1.2 Position Velocity Map on a 2D region

In this way, the position velocity map is a view for a 2D region on the sky of the relative speed of different portion of the object:

For each pixel the relative speed, computed from the Doppler effect, come from the maximal intensity of the local spectrum compared to the reference layer (or wavelength or speed) (Averaged position of the maximum ? Raw position of the maximum for a given position ? Reference coming from a reference layer ?)

The result of this is an image for which the color scale go from blue to red. Blue shifted to red shifted.

6.5.1.3 Common remark

6.5.1.3.1 Depth Dimension

Considering the description of the Cube, and the definition of this kind of map, the tool to compute a Position Velocity Map must be available only if the **dimension** for the the Z axis correspond to a (or spectral unit, and if the images are aligned on the sky, which should always be the case in a SimpleCube.

The resulting histograms and spectrum can be large if there is a large number of layer in the cube.

- For PACS, there is 2 mode of observation: Line-Spec and Range-Spec.
  - The Line-Spec correspond to an observation on a fixed position of the grating and result in a small domain of wavelength (~10 µm ??), which cover usually **one** line of emission, in a relatively limited number of "layer"

    One observation in Line-Scan mode can contain many elementary scan modes and have various spectrum separates the one from the other. For the Spectrum analysis
  - The Range-Spec correspond to the complete range of wavelength, using many positions of the grating. The result of this mode is usually a complex spectrum with many line of emission and therefore many local maximum. The resulting cube can contain a big amount of layer.

- For the other instruments the number of spectral values can also be huge.

6.5.1.3.2 *extension in depth*

In the first release this feature should be used only with cube constructed on a "restricted" range of wavelength (a specific emission line ...) because the definition of the relative velocities is made on the "shifting" of the position of the maximum of the spectrum.

If the spectrum is complex with many line of emission the computation of the relative velocities must be done via a correlation which is more complex and therefore be implemented latter, if needed.

An additional "feature" will be to consider only a given (sub)range of wavelength/frequencies from a range scan cube. therefore it will be possible to create a velocity position map without extracting a sub-cube with the dedicated tool (see §6.10.1).

- QUESTION:
  - For the "axis mode":

    due to the Number of spectral data compare (Y axis) to the number of pixel along the view of the velocity position map in a *PlotXY* component need to be formated, the aspect ratio must be modified. This can be done after to be shown but also from the GUI Before to display the result.

    We need input from the user to define the *default mode*.
  - For the 2 mode :

    the color scheme will use a common "color table" from blue to red: this can be change manually in the right button menu in the PlotXY interface (see below)

6.5.2 Features

6.5.2.1 Position Velocity Map along an axis

The final result for This kind of view is a 2D array (displayed as an image) with as dimensions:

- X = The number of pixels along the axis
- Y = The number of value on the spectral axis in the cube (Nb of layers)

The process can be summarized:

- Define a reference layer (i.e a reference spectral value) for the speed.
- Define a slit by defining an axis in the window and a width (a field in the GUI, one pixel by default)
- For each (set of) pixel(s) along the axis reading the "Z" axis and constructing an histogram of the intensities:
  - extraction of the raw(s) spectrum
    - if the width is >1 averaging the spectrum
  - (optional ?) smoothing to identify the unique maximum .
  - conversion in velocities.
- Display these histogram side by side in a resulting 2D image.

The main work of this tool is therefore to read the images along the spectral axis to extract the spectra, and convert them in relative velocities.

- QUESTIONS:
  - The default Smooth applied on the spectra is a Gaussian filter and the width is the Resolving Power of the instruments.
    - This value is not unique the tool **need to access to the Calibration Files**
  - If the users needs other filter or a Fitt this must be asked

6.5.2.2 Position Velocity Map on a 2D region

The result of this tool is for each pixel a couple of scalar : a velocity and a dispersion (or a FWHM). therefore the format of the result will be a set of 2 map.

6.5.2.2.1 Velocity map

For a line spec, the Velocity is related to the wavelength of the maximum of emission on line spec, which is compared to a given wavelength. the raw spectrum need therefore to be smoothed (filtered) or fitted .. depending of the user needs and the type of signal.

For a range spec the determination of the velocities need to determine the global shift of a complex spectrum, it need therefore more complex tools like autocorrelation. it therefore don't be applicable it the first versions of this tool.

For a quick analysis toolbox we propose to offer only velocity maps on line spec, and we can summarize the process:

- Define a reference layer (i.e a reference spectrum value) and store the λ_0 or f_0 in case of frequencies
- For each pixels
  - Extract the raw spectrum
  - Smooth the raw spectrum
  - Find the λ or f of the maximum of emission
  - Compute the relative velocity using the applicable Doppler formula $\Delta V = c\frac{\Delta\lambda}{\lambda_0}$ or $\Delta V = c\frac{\Delta f}{f_0}$
  - If wanted find the FWHM of the resulting line of emission by fitting a Gaussian (Default mode) on the raw spectra (centered on the position of the maximal value).
- Construct the map of the velocities and store it in a “composite dataset”

The final result will be a 2D array, containing the relative speed (the speed value for each pixel).

The value of the λ_0 is stored in some metadata

6.5.2.2.2 Dispersion map

Since for the Line-Scan the spectrum correspond to an unique line of emission it present an unique maximum. The “Dispersion Map ” is therefore is the map of the FWHM of this line, pixel per pixel.

The value of the width of this line of emission is the one resulting of the “*Gaussian fitt*” of the raw spectrum (or an other function to be define), pixel per pixel. (see above).

This tools won't be available for Range-Scan at this date

Remarks:

- **Since the computation of the value of the dispersion is made on the spectrum this calculus must be done at the same times as the the extraction of the spectrum.**

- The dispersion map should be accessible via a check box in the parameter area of the display, when existing.

The Final result will be stored in a SimpleCube or a SimpleImage.

6.5.3 Interface

The computation of the position velocity map is done in a Task “***VelocityPosMapComputeTask***” (to be renamed) for the 2 modes. This task need some parameters depending and these parameters can be initialized from a JIDE command line or in a GUI, the needed parameters depend of the kind of map the user need.

6.5.3.1 Axis mode

In this mode the position velocity map is mainly a view of a cut in the cube along a given plan with a specific scale-color related to the speed.

To define a plan we need two axis, and in our case:

- The first is always the same, the Z axis. with a control on the Spectral unit.
- The second one is defined in an image, the definition of this axis could be done by:
  - Its direction and a point of origin
  - Its equation.
  - The starting and ending points in image coordinates or in sky coordinates.

  The Task *VelocityPosMapComputeTask* take a double2d array defining the (X,Y) coordinates of the pixels to be read. the “selection” of the axis must be done in the GUI itself or from the command line via a computation.

  In the graphical interface different methods can be used to initialize this array:

  - drawing a line on one layer of the cube with the mouse
  - filling some “text field area”: first and last pixel : this will compute the equation of the line and fill the array

  If the line is drawn on the image this line is converted in coordinates, if the parameters are entered in the text field area, a corresponding line should appear in the image. and in both case the array of pixel is filled

To compute the relatives speeds we need a reference layer. For the selection of this layer we propose many solution:

- Selection via the slide bar of the cube visualization window. (to be done in the Display component)
- Manual input in a textfield area (this field should be automatically fill with the selected layer in the default mode)
- Automatic definition (layer of the maximal value of the spectrum at a given pixel (starting point of the line...)). (won't be implemented in the first version).

The final result of an "Axis mode" Position Velocity Map must be Displayed in a graphical tool so the Task proceed only to the computation of this map, based on the input parameters. The output parameters of the task are to be put in a simpleImage. The usage and the Tasks and the output depend of the mode of use.

- *Command line (call of the task):*
  - The task *VelocityPosMapComputeTask* take the following parameters :
    - "cube" as Double3d // the cube , Mandatory
    - "wave" as Double1d // wavelenght of the layers
    - "freq" as Double1d // frequencies of the layers
    - "axis" as Boolean // flag for the kind of usage: "map" or "axis", mandatory TRUE for the Axis mode
    - "coordAxisarray" as Double2d // array of the pixel to be read in the axis mode
    - "referenceLayer" as Integer // the layer of reference to be used for the spectral unit
    - "sizeOfSpectrum" as Integer // the inside size of spectrum (or number of layer of the cube)
    - "nbpixels" as Integer // the inside nb of pixel to be read
    - "dimX" as Integer // the inside X dimension of the cube, Mandatory
    - "dimY" as Integer // the inside y dimension of the cube , Mandatory

  The output parameters are:

    - "velocityMap"Double3d.class);// The 3D map of velocity, containing the velocities on one layer and the FWHM (NULL in the axis mode)
    - "velocityMapAxis" as Double2d // The map of velocity along the axis (NULL in the map mode)
    - "sigma" as Double // The set of parameter of the free gaussian fitt used for the definition of the position of the maximal value
    - "posMax" as , Int1d
    - "maxValue" as Double
    - "returnStatus" as Boolean

Display window:

  - In the main window the user select the mode "axis"
  - He select the reference frame (via the slide bar or a text field) and in this frame the starting pixel of the axis. From this starting pixel he draw the axis.
  - By default the reference will be the frame used for the definition of the axis. a textfield area in the right side of the window will show the wavelength of the selected layer.

- The task executing the computation is launched by clicking on the "compute velocity map"

6.5.3.2 2D map mode

The 2D map mode need only one parameter, the reference layer or wavelength. The choice of this layer could be done *via* 2 way:

- A direct definition via the slide bar or a textfield area
- A "computation" by finding the layer of maximal emission for a given pixel.

At the date of this document the Task used for the 2 types of Position Velocity Map is the same "*VelocityPosMapComputeTask"* so the most part of the parameters is the same.

- *Command line (call of the task):*
  - "cube" as Double3d // the cube , Mandatory
  - "wave" as Double1d // wavelenght of the layers
  - "freq" as Double1d // frequencies of the layers
  - "axis" as Boolean // flag for the kind of usage: "map" or "axis", mandatory TRUE for the Axis mode
  - "referenceLayer" as Integer // the layer of reference to be used for the spectral unit
  - "sizeOfSpectrum" as Integer // the inside size of spectrum (or number of layer of the cube)
  - "nbpixels" as Integer // the inside nb of pixel to be read
  - "dimX" as Integer // the inside X dimension of the cube, Mandatory
  - "dimY" as Integer // the inside y dimension of the cube , Mandatory

  The output parameters are:

  - "velocityMap"Double3d.class);// The 3D map of velocity, containing the velocities on one layer and the FWHM (NULL in the axis mode)
  - "velocityMapAxis" as Double2d // The map of velocity along the axis (NULL in the map mode)
  - "sigma" as Double // The set of parameter of the free gaussian fitt used for the definition of the position of the maximal value
  - "posMax" as , Int1d
  - "maxValue" as Double
  - "returnStatus" as Boolean

6.5.3.3 Selection axis-map mode (Graphical interface)

The choice between the 2 selections mode should be done in the graphical interface. the radio button define the value of the boolean trigger for the computation. and each of these mode will launch the computation process with the needed parameter and the reference layer.

6.5.4 Evolution , proposition for additional feature

The first "release" of this position velocity map , can only process a "single line" spectrum, so it can be run on "range scan" cube.

In order to use this feature for range scan it it will be necessary to "extract" single line or to define tools to compute the relative velocities from multi line cube. this will be a new feature (see §6.10.1).

6.5.5 Compatibility

Like all the other, this feature is designed to work on cubes, with sky aligned images. It will therefore be compatible with all the data which could be given to the main GUI if the are compliant with the cube specification.

Some of the features (or part of) defined in this tool won't be relevant for all the cube (feature for small range ...). All the "specialized" feature must be accessible only when they will be relevant.

6.5.6 Requirement on input data. (obsolete)

In order to work correctly the tool need a metadata in the incoming data, defining:

- The type of scan (Line-Scan Range-Scan)
- For the Line-Scan the number of scan

In the definition of the cube the wavelength should be available for all the pixel and all the layer. this point has to be checked and we need to have access to these value.

- QUESTIONS:
  - The velocities are computed with the "Simple" Doppler formula (non relativistic) if some user need to apply a more complex formula they must ask for it with a precise requirement.
  - We need information about the UNIT for the velocities : at this date the velocities will be displayed in Kilometers per seconds.

6.6 Generate Channel maps from a CubeDataSet

6.6.1 Definition of a channel

The definition of a channel, coming from the radio astronomy and millimeter astronomy is a, "band of wavelength, related to the internal speed of the observed objects". It's a vision of the dynamic of the sources.

A Channel is a false color view of the mono wavelength images of the object: The color of the pixel for a given wavelength is related to the intensity of the pixel at this wavelength. The channel map can be also shown as contour line.

Due to the definition of a channel this functionality will be available only if the data contain "images sky" where the unit of the "Z axis" is a spectral type. it mean that data must be compatible with the cube specifications.

6.6.2 Compatibility of the feature

Since this tool won't be useful on cube with a large amount of wavelength it will be available directly for the Line-Scan data. Since the computation of the relative speed on Range-Scan can't be done in the same way than for the Line-Scan, the Range-Scan data must be prepared, this will be explain latter.

6.6.3 Display of the channel map

6.6.3.1 Feature

The channel tool will have to show a “mosaic” of the individual wavelength images in the band with in the legend:

- The reference wavelength for the “0 Km/s” velocity
- The wavelength of the image
- The velocity corresponding to this wavelength.

The images in the mosaic will be displayed in false color images or contour level, depending of the user.

Since the users could prefer or work with other units than wavelength or velocities, we should think to a possibility to display channel map in additional units like wavenumber or frequency. The display and computation itself will depend of the unit so the choice of the unit should be done at the initialization of the tool.

The number of level can be define by the user in many way: For this we re-use the work done by S.Regibo on the image analysis toolbox.

Since the number of wavelength in a cube can be large, the interface won't display by default a mosaic of the channel maps. They will be shown frame per frame with a slide bar to navigate in the cube, in place of the original cube. A scale color will be computed, based on the level contour (TBD).

This tool should offer to save the set of images as *postscript* or other graphic format (To be define). At this date the choice are :

- Postscript
- PNG images

The idea is also to offer a print tool which could offer many format, (To be defined). At this date we propose:

- Print the images one by one
- Print all the images in a mosaic (with a manual set up)

6.6.3.2 Interfaces

In order to apply test harnesses on the different part of this tool, most of the methods should be available via a “command line” interface and the GUI will only interact with the user construct the parameters send them in the good format to the various method and recover the data to send them in the graphical display area. The basic operation are essentially some “graphical aspect” tools:

- Computation of contour values.
- Modification of scale color
- Drawing of contours ...

The computation of the contour values and of the contours them self can be made in a separate class, but everything depend of the cube and should be display so for this feature it won't be possible to completely separate the various part of the computation, at least some data must will have to be global, and shared with different classes.

Nevertheless for the computation part itself we can define a "command line" process and therefore the interface for this tool will be.

- *Command line:*
  - The arguments will be the reference layer (as Lambda or number), the bandwidth (in lambda), the conversion rule from lambda to relative speed. An optional command could be the display mode (contour, color).
  - the default output will be a set of levels and a new color scale. A specific method will prepare the print of the mosaic or the set of images to save as graphic file.
- *Display:*
  - The display interface will only launch the command line tool after assembling the parameters from graphical entries (text area and check box) and recover the contour levels and color scale to apply it to the cube, from the output of the command line class.
  - The channel mosaic will be displayed in the main window and a radio button will allow to switch from the false color to the contour map.
  - The displayed mosaic will be savable as fits file or graphic file via a file/save menu. (need to decide where to put the "save as file " tool.)

6.6.3.3 Requirement on input data

The main need for this tools is a control on the unit of the Z axis (wavelength or "spectral" unit). All the cube should have a metadata to inform of this unit.

6.6.4 Preparation of the cube for the channel map.

6.6.4.1 Introduction

For PACS the Line-Scan mode will deliver "small" scans easy to manage, but the Range-Scan mode will deliver some very big scans with a large amount of layers. The other instruments could also deliver some voluminous spectra (?)

The Channel tool is not adapted to the very voluminous spectra and for this kind of cubes it is necessary to reduce the number of layers to be displayed.

A specific tool available as first part of the channel map tool should allow "reduce" the cube. This reduction tool will mainly propose to combine (by an integration or an average) many layers in one and associate to this resulting layer a central **wavelength and a bandwidth.**

Since the **bandwidth** is not present in the cube, the bandwidth for the Line-Scan will be the distance in wavelength unit between to layers.

6.6.4.2 Features

<u>Line-Scan:</u>

For the computation of the cut level the user needs information about :

- The number of layer

- The range of wavelength
- and in “real time” the range of corresponding velocities.
- The bandwidth for the layers

These information must be displayed in a specific area.

From these information the user can define

- The number of cut level
- The way to compute them (automatic, linear log ...)
- A list of cut level by typing them manually

Range-scan (and large spectra data)

For the data presenting large number of wavelength or frequencies (or layers), a first tool must allow the user to reduce the data. this tool should give to the user the possibility to group some layers in a “averaged” or “combined” layer.

The user will have to decide the final number of layer or the final bandwidth. When one property will be chosen the other one will be computed. We propose that the bandwidth is constant for all the spectrum.

The method to reduce the cube should also be selectable:

- Simple sum
- Weighted average ...

Finally after the definition of the parameters in the GUI, the cube will be processed and a new one will be created.

For the Graphic interface, the original and the final cubes could be displayed, in order to compare the result of the preparation. The final implementation will depend of the requirement of the users during this specification process.

The channel visualization will be applicable only on the new “reduced” cube. Therefore we need a flag which will help to define a cube compatible for the channel tool, this flag will be:

- **True** for the Line-Scan
- **False** for the raw Range-Scan
- and **True** for the processed Range-Scan

Finally for Line-Scan cube there will be 2 cubes displayed (as 2 tag on the left side ? or the second one in a display section under the parameter zone on the right side). the channel map will be applicable only.

6.7 Line-fitting package on all elements of a CubeDataSet

6.7.1 Introduction

The Line-fitting is a basic tool which is used by the other tools presented above. Therefore the line-fitting package for this GUI will be a toolbox containing various method to fit Spectra and/or images with various algorithm. This package will have to be compatible with 1D (spectra) data. Even if it's a **Line-**fitting package it could be interesting to investigate the possibility to apply these tools on 2D (layers or mono wavelength images) data.

The Line fitting is a common tool for all kind of spectra or 1D data. so in order to avoid double development we looked for some fitting libraries already available or in development in the PCSS.

for this libraries we looked also if they were (or if they could be "modified" in order to be) compatible with the cube format specification.

At this date we identified:

- The "Line Fitting Tool for Herschel" developed by Rob Assendrop (SRON) which is described in the document **FitterRequirement0.5.**

  We'll work in collaboration in order to use compatible data type and to find in this libraries
- Some other libraries in herschel.ia.numeric.toolbox.fit developed for PACS SPIRE or HIFI. We'll continue to investigate the classes available to see which could be used.

The next step of the Wps, named "Create map from fitted parameters", an other set of tool contained in the Features described above, will use the result of this toolbox as part of its input, in particular

for the 'task'

- Create a new CubeDataSet containing the result of the fits and/or the filters
- Save the resulting CubeDataSet on disk
- Return the new Cube to the process which called the tool (IA specific GUI...)
- Eventually, when applicable, save a set of "numbers" which defined the fit and which will be read at the next point to create the specified map from this step.

Therefore we can consider that this toolbox only need to be a set of filter and or "mathematical function" to apply on some part of the cube. Since we consider that we'll re-use on of the HCSS libraries listed above, we need a exhaustive list of the fits the user want.

If some fits are missing in the chosen library we'll ask the the responsible of the package to add them.

For the Line-fitting, the results will finally be stored in some **Spectrum1D,** if the output format is Float1D or Double1D or set of arras (or other) these data will have to be converted in Spectrum1D.

6.7.2 Interfaces

the line-fitting apply on a spectrum (basically 1D array with wavelength information) and it return a new spectrum. when applied on a complete cube it will be use in a loop. so the basic interface will take:

- a spectrum
- a set of parameters ..

and will return a spectrum.

Since we'll use the line-fitting toolbox developed in SRON we, as user of the toolbox, will send them a list of required fit process and input output format. actually the idea is to work in collaboration on this toolbox and discuss with them in order to converge.

For the features using line fitting, the GUI will launch the selected process with the needed parameters and spectra, then it will receive the resulting spectrum/cube.

The GUI will allow to save the results as

- CubeDataSet
- Cube (compatible with the input of the *cubeSpectrumAnalysistoolbox* in order to re-inject this result in the GUI to display its property)
- Float3D with wavelength associated
- Spectrum1D or Dataset containing spectrum1D
- *... Line List Product (Do Kester).*

The GUI should also return the result to the environment from where was launched the *cubeSpectrumAnalysistoolbox*.

6.7.2.1 propositions

For all the feature which will use some line-fitting possibilities in the right part of the window there must be

- Some information about the cube:
  - extension range in wavelength
  - coordinates of the center of the cube
  - the number of layers and the step between these layers.
  - eventually a clear flag if the cube is a **Line spectrum** or a **Range spectrum** (for PACS data)
- A list of fit available with for each of them a sub section for the parameter.
  - a fit applicable along the Z axis (like a Gaussian fit) should be applicable with the same conditions on each pixels and therefore an option to apply it on all the spectra or only on one will be available.. For this type of fit, at this date identified
    - Gaussian
    - Voigt profile
    - Sinc function (*sin(x)/x)* ...
  - A fit applicable on one image should be applicable with the same conditions on each layers. For this type of fit, at this date identified
    - Gaussian
    - Deconvolution by the PSF (??)
- A section to save the resulting fitted spectra and or cube in various formats:
  - for the cube:
    - Cube dataset "HCSS" compatible
    - Fit table
    - 3D arrays
  - for the individual spectra.
    - Spectrum 1D or 2D dataset "HCSS" compatible
    - Fit table
    - ASCII or text file
    - 1D or 2D arrays
  - Values for the Jide environment
    - internal spectrum type
    - new cube ...

6.7.3 comments about arithmetic on spectra

In addition of the line-fitting package, some arithmetic could be done in the GUI on the individual spectra. From the PACS cube the basic operations like continuum extraction will be done in the CubeBuilder module, but for the spectra coming from the other instruments or for some very specific arithmetics filter or conversion operations we need a dedicated interface.

A first list of filter and processing tools can be define like.

- Continuum extraction
- Arithmetics on spectrum
    - subtraction of line ...
    - addition of line
    - division, multiplication
    - Conversion from wavelength to frequency ...

Since for PACS all these operations should be done at the previous step, if there is a need to keep an “arithmetic interface” for very specific operation and for the other instrument. the users MUST deliver a precise list of the various operation needed.

Without any feed back from the user on this point nothing will be implemented.

If implemented, the arithmetic interface will use the library developed by **Martin Melchior** in Zürich and located in the HCSS at : Package herschel.ia.toolbox.spectrum.arithmetics

6.8 Create map from fitted parameters

6.8.1 Inputs and features

The input data of these tools will be some output of the previous point. The idea is to offer Display for fitted/filtered cubes.

“map from fitted parameters” are maps constructed not from the fitted signal themselves but from some values resulting of the fitting process.

for example in the case of a Gaussian fit of the individual spectrum the fitted parameters are the wavelength of the maximum the maximum value of the Gaussian and the sigma value, related to the dispersion. In this case the *maps of the fitted parameters* will be the *position velocity map*, and the *dispersion map*.

therefore this WP is already, for most of the identified needs, described at the beginning of this document. if things are missing we need feed-back from the users.

6.9 Create maps based on observed line properties of individual spectra

6.9.1 Inputs and features

In the same idea that for the previous paragraph the “creation of maps based on observed line properties of individual spectra” take some values and or arrays coming from the fitting process. combine them and display the result. therefore this WP is already partly described at the beginning of this document. If things are missing we need feed-back from the users.

It seems that a real 3Dimension visualization tools could be useful for some users. in this case this is a kind of Create maps based on observed line properties of individual spectra of a complete cube. This is a complex tool and if there is a demand for it will be developed latter.

To define such a kind of tool see the TOPCAT software from the ESO.

6.10 Proposition for additional features

6.10.1 Range extraction

This should be a basic GUI for extracting a range of wavelength from a complete cube.

the interface could:

- Display an information field about the spectral range.
- Display an averaged spectrum on the whole image. to identyify the "spectral content"
- activate a selection tool to define the spectral region to ectract (extension of the range to extract). This definition would be done graphicaly by selecting a "band" with the mouse
- Display an "extract button" which create a new cube on the range selected and activate a "save data" button or a "return data" button
- Display some information on the resulting cube.

The meta data of the resulting cube would be copied from the original one, with an update on the spectral information.

This GUI would call a Tasks which would take

- a cube
- a range

and extract the new cube and put it in a new SimpleCube

6.11 Main window

Like said before, the various tools will be available from the same main window. There will be some display tools but some of the "fit and computation" tools could also be available without graphical output (for the "preparation" of the cubes or the last operation). Therefore the main window should present 2 main entries in the menu, one for the operation to be done with a graphical result, one for the fits and the computation which will return only products.

The "fits" tools and all the ones which create new filtered cubes will save the new result and therefore it should be possible to load new cube from file from a "file" menu in the main window.

On October 2007 a group is working on the re definition of the GUIs architecture internal communication and so on. the Final main window will depend of the conclusion of this group and therefore the final development for the GUI itself will be done when these conclusions will be available. the "cubespectrum_analysis_toolbox_Develop_requirement" document, available at the same place in the wiki pages, explain the internal architecture for the global window and feature per feature.