User Requirements for Creating and Processing a Spectral Cube from SPIRE data

SPIRE-BSS-DOC-003175

Author:
Peter Davis-Imhof

Issue:
0.3

Date:
June 2, 2009

Document History:

Date	Version	Comment
February 7, 2009	Issue 0.1	For internal review
February 17, 2009	Issue 0.2	For ICC review
June 2, 2009	Issue 0.3	For ICC aceptance (Basic Functionality)

Table of Contents:

A. Reference Documents 2
B. Document Scope 4
C. Objectives 4
D. Use Cases for the Basic Functionality 4
1. CubeCreation - CreateJiggleCube via the pipeline 4
2. CubeCreation - CreateJiggleCube interactively 4
3. CubeCreation - AverageSinglePointing onto grid 5
4. CubeProcessing - Regrid a regular spectral cube (spatially) 5
5. CubeProcessing - Basic resampling of a regular spectral cube (spectrally) 5
6. CubeProcessing - Combine two regular spectral cubes 5
E. Use Cases for the Advanced Functionality 5
1. CubeCreation - CreateRasterCube via the pipeline 6
2. CubeCreation - CreateRasterCube interactively 6
3. CubeProcessing - Convolve a regular spectral cube to a user-defined Point Spread Function (PSF) 6
4. CubeProcessing - Advanced resampling of a regular spectral cube (spectrally) 6
F. Approach to creating spectral cubes 6
G. Detailed Requirements for Basic Functionality 7
1. SpireCubePreProcessingTask 7
2. SpectralProjectionTask 9
3. ResampleFrequency 12
H. Detailed Requirements for Advanced Functionality 12
1. Cube creation from a raster observation 12
2. ResampleFrequency 12
3. PsfConvolutionTask 12

A. Reference Documents

RD-1	SPIRE Spectrometer Pipeline Description, SPIRE-BSS-DOC-002966, issue 1.2, December 5, 2008
RD-2	A Basic User's Manual – Scripting in the Herschel Data Processing System, HERSCHEL-HSC-DOC-0517, version 0.24.1730, October 14, 2008
RD-3	SPIRE Operating Modes for the SPIRE Instrument, SPIRE-RAL-DOC-000320, issue 4.0, August 1, 2008
RD-4	SPIRE Pipeline Mask Policy, SPIRE-BSS-DOC-003127, issue 1.3, January 28, 2009

RD-5	Mask and error handling in the PACS spectrometer pipeline, user requirements, 4th draft, October 16, 2008

B. Document Scope

This document defines the requirements for creating and processing an equidistantly sampled, spectral cube from data taken by the SPIRE imaging Fourier transform spectrometer (FTS). This document is structured as follows: **Section C** defines the objectives of the software development. **Sections D** and **E** define the basic and advanced use cases, i.e. high-level scenarios for users to create and process spectral SPIRE cubes. **Section F** explains the overall approach to providing the required functionality. **Sections G** and **H** give detailed requirements, traceable though the Herschel Change Management System, for the basic and advanced creation and processing of spectral SPIRE cubes.

C. Objectives

The cube creation should lead to a level-2 data product, i.e. one that is – as much as possible – free of the characteristics of the SPIRE FTS. It should allow astronomers to quickly derive scientific results from the hyperspectral data, working with maps at certain wavelengths (line positions) and individual spectra within the cube. It is therefore required to derive a "regular" spectral cube, i.e. one with equidistant spatial and spectral grids. A regular spectral cube will also facilitate comparing SPIRE data to data from PACS and HIFI, the other two science instruments on Herschel, and other astronomical observatories.

D. Use Cases for the Basic Functionality

The uses cases in this section define the scope for the ESA extension work package 6.6.1.1. SPIRE FTS to Cube (Basic Module). The use cases relate to pipeline and interactive processing of SPIRE spectral cube data. Use cases 1 to 3 apply to the creation of a SPIRE spectral cube. Use Cases 4 to 6 apply to the processing of a SPIRE spectral cube.

1. CubeCreation - CreateJiggleCube via the pipeline

Create a spatially regularly sampled cube from a jiggle map observation, identified by one obsid via the automated data processing pipeline.

a) Select all the SpectrometerDetectorSpectrum products from 4/16 building blocks of a spectral jiggle map when processing the data from one observation. [TBD: non/apodized spectra]

b) Apply a SpectralProjectionAlgorithm (e.g. PACS projection or HIFI regridding).

2. CubeCreation - CreateJiggleCube interactively

Create a spatially regularly sampled cube from a jiggle map observation, identified by one obsid via interactive analysis.

a) Select all the SpectrometerDetectorSpectrum products from 4/16 building blocks of a spectral jiggle map for a given obsid.

b) Define the target sky positions as an equidistant rectangular grid (optional user input: dx=dy, RA of bottom left corner, DEC of bottom left corner, N_RA, N_DEC, angle ?)

c) Apply a SpectralProjectionAlgorithm (e.g. PACS projection or HIFI regridding).

3.CubeCreation - AverageSinglePointing onto grid

Average a sparsely sampled, single pointing observation onto a regular spatial grid.

a) Select the SpectrometerDetectorSpectrum product from one building block of a spectral point observation. [TBD: non/apodized spectra]

b) Define the target sky positions as an equidistant rectangular grid; required user input: dx=dy, RA of bottom left corner, DEC of bottom left corner, N_RA, N_DEC, angle ?

c) Apply a SpectralProjectionAlgorithm (e.g. PACS projection or HIFI regridding).

4. CubeProcessing - Regrid a regular spectral cube (spatially)

a) Load a regular spectral cube into memory

b) Define the target sky positions as an equidistant rectangular grid; required user input: dx=dy, RA of bottom left corner, DEC of bottom left corner, N_RA, N_DEC, angle ?

c) Apply a SpectralProjectionAlgorithm (e.g. PACS projection or HIFI regridding).

5. CubeProcessing - Basic resampling of a regular spectral cube (spectrally)

a) Load a regular spectral cube into memory

b) Define the target wavescale grid; required user input

c) Spectrally resample each spectrum in the spectral cube with a standard interpolation routine

6. CubeProcessing - Combine two regular spectral cubes

a) Load two regular spectral cubes into memory

b) Define the target wavescale grid; required user input

c) Spectrally resample each spectrum in the spectral cube

d) Define the target spatial grid; required user input: dx=dy, RA of bottom left corner, DEC of bottom left corner, N_RA, N_DEC, angle ?

e) Apply a SpectralProjectionAlgorithm (e.g. PACS projection or HIFI regridding).

E. Use Cases for the Advanced Functionality

The uses cases in this section define the scope for the ESA extension work package 6.6.1.2. SPIRE FTS to Cube (Advanced Module). The use cases relate to pipeline and interactive processing of SPIRE spectral cube data. Use cases 1 and 2 apply to the creation of a SPIRE spectral cube. Use cases 3 and 4 apply to the processing of a SPIRE spectral cube.

1. CubeCreation - CreateRasterCube via the pipeline

Create a spatially regularly sampled cube from a raster map observation, identified by one obsid

a) Select all the SpectrometerDetectorSpectrum products from n times 1/4/16 building blocks of a spectral raster map when processing the data from one observation. [TBD: non/apodized spectra]

b) Apply a SpectralProjectionAlgorithm (e.g. PACS projection or HIFI regridding).

2. CubeCreation - CreateRasterCube interactively

Create a spatially regularly sampled cube from a raster map observation, identified by one obsid

a) Select all the SpectrometerDetectorSpectrum products from n times 1/4/16 building blocks of a spectral raster map for a given obsid.

b) Define the target sky positions as an equidistant rectangular grid (optional user input: dx=dy, RA of bottom left corner, DEC of bottom left corner, N_RA, N_DEC, angle ?)

c) Apply a SpectralProjectionAlgorithm (e.g. PACS projection or HIFI regridding).

3. CubeProcessing - Convolve a regular spectral cube to a user-defined Point Spread Function (PSF)

a) Load a regular spectral cube into memory

b) Define the target point spread function (as a function of wavescale); required user input

c) Convolve the layers of the spectral cube with the target PSF

4. CubeProcessing - Advanced resampling of a regular spectral cube (spectrally)

a) Load a regular spectral cube into memory

b) Define the target wavescale grid; required user input

c) Spectrally resample each spectrum in the spectral cube with an optimized procedure involving interpolation via Fourier transformation, zero-padding, inverse Fourier transformation, and linear interpolation.

F. Approach to creating spectral cubes

The approach to providing the functionality for the basic cube creation is detailed in Table 1 and Figure 1:

1	Access data either via the pipeline or the PAL browser
2	Pre-process spectral products from SPIRE to convert data into a generic format
3	Apply a spectral projection task (shared between Herschel's science instruments) to create a regular cube

Table 1: Sequence of the cube creation steps

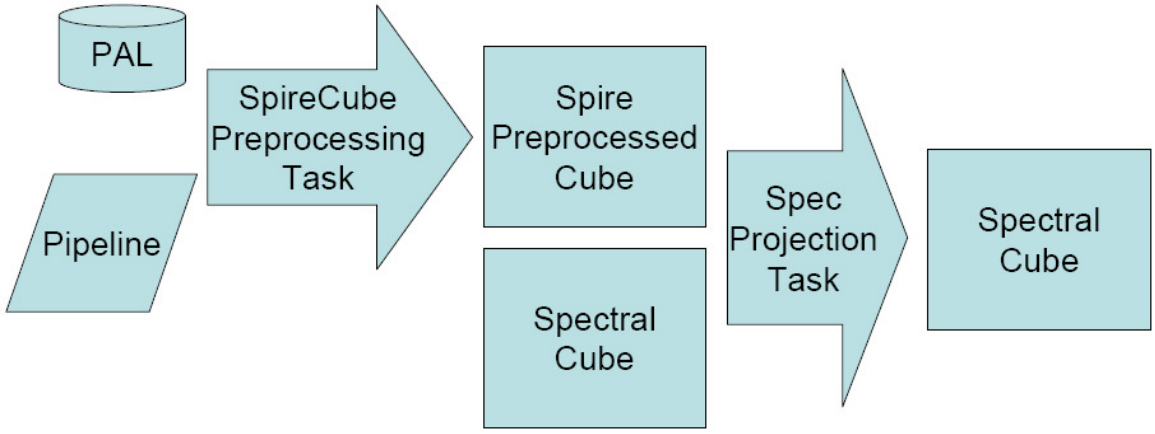

Figure 1: Data flow for the cube creation

G. Detailed Requirements for Basic Functionality

The requirements here define the scope for the ESA extension work package 6.6.1.1. SPIRE FTS to Cube (Basic Module).

1. SpireCubePreProcessingTask

The implementation of the SpireCubePreProcessingTask will provide the core functionality for the second step outlined in Table 1 and will cover use cases D.1. – D.3. A respective SPIRE SCR-1262 has been raised.

a) Input Data

CPP-1 The SpireCubePreProcessingTask shall operate on an ArrayList of SpectrometerDetectorSpectrum products from a Level1Context.

Comments:

- This will allow for automated cube creation: The SOF1 Point Source and SOF2 Field Mapping (see RD-1 and RD-3) pipelines will select suitable products (SpectrometerDetectorSpectrum with real spectra of suitable resolution (H+L AOT!), either apodized or not apodized) and pass them into the SpireCubePreProcessingTask.
- This will allow for interactive cube creation: A user can select suitable observations via the browser tool for the Product Access Layer (PAL) of the Herschel Common Software System (HCSS), see RD-2, chapter 12.1.10. This user interaction will result in an ArraySet of products, which can then be passed into the SpireCubePreProcessingTask.
- The assumption is that the sky position is attached to each individual SpireSpectrum1d in the SpectrometerDetectorSpectrum product. Spectrum1d has been updated for build 0.6.7. SpireSpectrum1d will be updated for build 1.0.
- We plan to ignore the differences between the instrumental line shapes (ILS) for different detectors as a first step. The ModulationEfficiencyCorrection task should be

implemented to correct for these differences. This task can be executed as part of the pipeline or possibly applied to the pre-processed cube.

CPP-2 The SpireCubePreProcessingTask shall verify that the wave column is identical within a small margin of 1e-14 for all detectors within a given scan of a SpectrometerDetectorSpectrum product.

CPP-3 The SpireCubePreProcessingTask shall verify that the wave column is identical within a small margin of 1e-14 for all available scans of a SpectrometerDetectorSpectrum product.

CPP-4 The SpireCubePreProcessingTask shall verify that the wave column is identical within a small margin of 1e-14 for all input SpectrometerDetectorSpectrum products.

Comments:

- A High + Low observation will provide different but compatible wave columns in the sense that each element in the wavescale of the low resolution observation will also be contained in the wavescale of the high resolution observation. However, before processing data from such an observation, it will be beneficial in terms of SNR to resample the high resolution observation onto the wavescale of the low resolution observation.
- It is not necessary to verify that the column lengths within one SpireSpectrum1d are identical since SpireSpectrum1d extends a StrictTableDataset which enforces equal column lengths. NB: This will be in effect only with the implementation of SPIRE SCR-0817.
- It is not necessary to verify that the lengths of the flux columns are of equal length because it is implied by the fact that the wave columns are identical.
- If any of the data consistency checks fails, then the task will issue a meaningful log message detailing the problem as much as possible and throw a signature exception, i.e. not run to completion.

b) Output Data

CPP-5 The SpireCubePreProcessingTask shall create a data product (SpirePreprocessedCube) which provides the input for tasks implementing the SpectralProjectionAlgorithm interface.

Comments:

- This requirement ensures a working interface to the projection algorithm.
- The interface is currently under review.

CPP-6 The SpirePreprocessedCube shall provide the data from all the columns of the original SpireSpectrum1d datasets (at least wave, flux, error, mask) for the detectors of

the SLW array only, from the detectors of the SSW array only, or from all SPIRE detectors.

CPP-7 The SpirePreprocessedCube shall issue a specific and meaningful log message if not all the requested 3D data structures (flux, error, RA, DEC, flag) have identical lengths.

CPP-8 The SpirePreprocessedCube shall issue a specific and meaningful log message if the length of the wavescale does not match the corresponding length of the requested flux cube.

CPP-9 The output data shall be sufficient to provide input for a task that implements the SpectralProjectionAlgorithm interface.

c) Functional requirements

CPP-10 The SpireCubePreProcessingTask shall only re-order the wave, flux, and error input data from the SpireSpectrum1d datasets to match the required output data format but not change the data values.

CPP-11 The SpireCubePreProcessingTask shall provide a flag that specifies – at a minimum – which flux samples are not to be used by the spectral projection task.

CPP-12 The SpireCubePreProcessingTask shall set a Boolean marker in the flag where the Master bit of a flux sample of the SpireSpectrum1d is set to True.

CPP-13 The SpireCubePreProcessingTask shall propagate those metadata entries that are common to all input products to the output data product.

Comments:

- Some information will be lost as part of the pre-processing (detector name, non-nominal detectors, scan number, scan direction, …).

2. SpectralProjectionTask

The implementation of a SpectralProjectionTask will provide the core functionality for the third step outlined in Table 1 and will cover use cases D.1. – D.4 and D.6. A SpectralProjectionTask is a task which implements the SpectralProjectionAlgorithm interface. The SPIRE SCR 1474 has been raised respectively.

a) Input Data

The SpectralProjectionTask shall provide three methods: project, beamProfile, and targetGrid. The project method shall be able to operate on three different types of data:

SPT-1 The project method of the SpectralProjectionTask shall be able to operate on an unprojected spectral cube as provided by the SpirePreProcessedCube:

Flux	Double3d	flux per beam
Ra	Double3d	Right Ascension
Dec	Double3d	Declination
Error	Double3d	error of the flux per beam
Flag	Flag (3d)	data quality flag
WCS	WCS (3d)	world coordinate system defining the target grid
allowExtrapolation	Boolean	non-mandatory indicator whether extrapolation be allowed

Comment:

- We also need a means to keep track of which detector recorded which spectrum if we want to use detector-specific beamProfiles.

SPT-2 The project method of the SpectralProjectionTask shall be able to operate on a regular spectral cube:

spectralCube	SpectralSimpleCube	
WCS	WCS (3d)	world coordinate system defining the target grid
allowExtrapolation	Boolean	non-mandatory indicator whether extrapolation be allowed

Comment:

- This will satisfy use case D.4 Regrid a regular spectral cube (spatially).

SPT-3 The project method of the SpectralProjectionTask shall be able to operate on a list of regular spectral cubes:

spectralCube[]	SpectralSimpleCube[]	
WCS	WCS (3d)	world coordinate system defining the target grid
allowExtrapolation	Boolean	non-mandatory indicator whether extrapolation be allowed

Comment:

- This will satisfy use case D.6 Combine two regular spectral cubes.

The beamProfile method shall be able to receive one set of input parameters:

SPT-4 The beamProfile method of the SpectralProjectionTask shall provide the normalized beam efficiency (integrated area/maximum value (TBC) is equal to 1) for a given detector (specified by an index/string (TBC)) as a function of deviation from the center of the detector in the two directions RA and DEC and the rotation angle (not mandatory).

The targetGrid method shall be able to operate on two different types of data:

SPT-5 The targetGrid method of the SpectralProjectionTask shall provide a default target grid from RA and DEC information:

Ra	Double3d	Right Ascension

Dec	Double3d	Declination

SPT-6 The targetGrid method of the SpectralProjectionTask shall provide a default target grid from an ArrayList of World Coordinate Systems.

b) Output Data

SPT-7 The project methods of the SpectralProjectionTask shall create a spectral cube, i.e. a 3D data structure with two spatial and one spectral dimension, as output product.

SPT-8 The project methods of the SpectralProjectionTask shall create a spectral cube specifying the flux per pixel (not per beam).

SPT-9 The beamProfile method of the SpectralProjectionTask shall return a number of type double between 0 and 1.

SPT-10 The targetGrid methods of the SpectralProjectionTask shall return a World Coordinate System with two spatial dimensions.

c) Functional requirements

SPT-11 The project methods of the SpectralProjectionTask shall set the flux and error at the target grid locations to the values of the closest input data (next neighbour interpolation).

SPT-12 The project methods of the SpectralProjectionTask shall create a spectral cube that contains all sky positions of the input data if the Boolean input parameter allowExtrapolation is set to True.

SPT-13 The project methods of the SpectralProjectionTask shall create a spectral cube that includes only those sky positions where data are available below and above the target position in both spatial dimensions if the Boolean input parameter allowExtrapolation is set to False.

SPT-14 The project methods of the SpectralProjectionTask shall set the Boolean input parameter allowExtrapolation to False as default setting.

SPT-15 The beamProfile method of the SpectralProjectionTask shall return the value of a Gaussian beam profile with FWHM of 16 arcsec/35 arcsec for detectors from SSW/SLW regardless of the rotation angle.

SPT-16 The targetGrid method of the SpectralProjectionTask shall create a World Coordinate System in two spatial dimensions with the reference pixel at the smallest RA and the smallest DEC values in the input data.

SPT-17 The targetGrid method of the SpectralProjectionTask shall create a World Coordinate System in two spatial dimensions with square pixel sizes.

SPT-18 The targetGrid method of the SpectralProjectionTask with an ArrayList of World Coordinate Systems as input shall create a World Coordinate System where the side of one square pixel is equal to the largest interval specified in any of the input World Coordinate Systems.

SPT-19 The targetGrid method of the SpectralProjectionTask with an RA and a DEC cube as input shall create a World Coordinate System where the side of one square pixel is equal to the difference between the largest RA and the smallest RA divided

by the square root of number of individual spectra present in the input data or the difference between the largest DEC and the smallest DEC divided by the square root of number of individual spectra present in the input data – whichever is greater.

SPT-20 The targetGrid method of the SpectralProjectionTask shall throw a signature exception if the input data straddle the origin of RA or the poles of DEC.

SPT-21 The SpectralProjectionTask shall ignore any data samples where the flux value is NaN or infty and not throw a Java signature exception.

SPT-22 The SpectralProjectionTask shall ignore any data samples where the master flag is set to True and not throw a Java signature exception.

Comments:

- The last two requirements follow RD-4, section 2.1.3, and RD-5, section 3.

3. ResampleFrequency

The spectrum toolbox (herschel.ia.toolbox.spectrum.ResampleFrequency) provides the functionality to resample the frequency grid of the spectral cube (satisfies use case D.5.) with two generic interpolation algorithms: an Euler integration scheme in combination with nearest neighbor interpolation and a trapezoidal integration scheme in combination with linear interpolation.

H. Detailed Requirements for Advanced Functionality

1.Cube creation from a raster observation

No new software is required in this case. The SpireCubePreProcessingTask and a projection task should provide the required functionality. However, it will be necessary to test the performance of these classes and adjust them if necessary. Performance Verification and Science Demonstration should provide suitable test data.

2. ResampleFrequency

The spectral cube class implements the interface SpectrumContainer. The ResampleFrequency task allows for a customized implementation of the resampling algorithm. This framework will be used to provide the advanced resampling method.

The advanced resampling algorithm is as follows:

1. Perform the inverse FT on the spectrum with N elements
2. Insert F · N zeros (F could be 10 and is tbd)
3. Perform the forward FT to compute a spectrum oversampled by a factor of F
4. Interpolate linearly onto the target grid

This covers Use Case E.4 CubeProcessing - Advanced resampling of a regular spectral cube (spectrally)

3. PsfConvolutionTask

A dedicated task will provide the functionality to convolve the slices from the cube to a user-defined PSF.

TBW

This covers Use Case E.3 CubeProcessing - Convolve a regular spectral cube to a user-defined Point Spread Function (PSF).