Module Interface Specification for SpectrumImageAnalysisPy

Isobel Bicket

December 17, 2017

1 Revision History

Date	Version	Notes
November 29, 2017	1.0	Initial draft

2 Symbols, Abbreviations and Acronyms

See SRS documentation at https://github.com/icbicket/SpectrumImageAnalysisPy/blob/SpectrumImageAnalysisPy_dev/Doc/SRS/SRS.pdf.

Contents

3 Introduction

The following document details the Module Interface Specifications for SpectrumImageAnalysisPy, a library created for the data processing of spectrum image datasets.

Complementary documents include the System Requirement Specifications and Module Guide. The full documentation and implementation can be found at https://github.com/icbicket/SpectrumImageAnalysisPy/tree/SpectrumImageAnalysisPy_dev.

All modules within SpectrumImageAnalysisPy are accessible by the user from terminal commands. Some modules will interface with each other, but the workflow of SpectrumImageAnalysisPyis driven by the user.

4 Notation

The structure of the MIS for modules comes from [1], with the addition that template modules have been adapted from [2]. The mathematical notation comes from Chapter 3 of [1]. For instance, the symbol := is used for a multiple assignment statement and conditional rules follow the form $(c_1 \Rightarrow r_1|c_2 \Rightarrow r_2|...|c_n \Rightarrow r_n)$.

The following table summarizes the primitive data types used by SpectrumImageAnalysisPy.

Data Type	Notation	Description
character	char	a single symbol or digit
string	str	a sequence of characters
integer	$\mathbb{Z}$	a number without a fractional component in (-∞, ∞)
natural number	$\mathbb{N}$	a number without a fractional component in [1, ∞)
real	$\mathbb{R}$	any number in (-∞, ∞)
complex	$\mathbb{C}$	any combination of real and imaginary numbers, in the form $a + bi$, where a and b are real and i is the imaginary number

The specification of SpectrumImageAnalysisPy uses some derived data types: sequences, strings, and tuples. Sequences are lists filled with elements of the same data type. Strings are sequences of characters. Tuples contain a list of values, potentially of different types. In addition, SpectrumImageAnalysisPy uses functions, which are defined by the data types of their inputs and outputs. Local functions are described by giving their type signature followed by their specification. It should be noted that each state variable is assumed to have a setter and getter accessible through module.variable_name.

5 Module Decomposition

The following table is taken directly from the Module Guide document for this project. The modules in this document are listed in the order in which they appear in this table.

Level 1	**Level 2**	**Level 3**
Hardware-Hiding Module		
Behaviour-Hiding Module	Import	csv dm3 h5 rpl
	Export	csv h5 png rpl
	Data processing	Richardson-Lucy Deconvolution Normalization Gain correction Background correction
	Data extraction	1D slice 2D mask 3D mask
	Display	1D spectrum plot 2D image plot 3D spectrum image plot
Software Decision Module	Data	Spectrum Image Spectrum Image
	Array Data Structure	
	Plotting Library	

Table 1: Module Hierarchy

6 MIS of Hardware Hiding Module

[You probably don't need to document this module, unless you need it to make the specification for other modules clear. —SS][You may be right, it's not needed, but I wanted to do it for completeness in not leaving out modules. I think writing this down also helped me to think about exactly what responsibilities I was passing to this module. It helped me, at least :) —Author].

6.1 Module

HardwareHiding (M1)

6.2 Uses

N/A

6.3 Syntax

6.3.1 Exported Access Programs

Name	In	Out	Exceptions
InputDevices	Hardware	Read out	-
OutputDevices	Hardware	Write out	-

6.4 Semantics

This module handles the interface between the hardware being used and inputs to the software

6.4.1 State Variables

N/A

6.4.2 Environment Variables

- Keyboard
- Mouse
- Screen
- Long Term Storage
- Temporary storage

[Yes, these are environment variables. There seems to have been some confusion on this topic, but you get it. :-) —SS]

6.4.3 Access Routine Semantics

InputDevices():

- input: Hardware allowing the user to input instructions to the computer software, *e.g.* mouse, keyboard, long term or temporary memory
- transition: N/A
- output: Software instructions corresponding to the desire of the user (*e.g.* registering a mouse click, reading a variable from memory, accessing a file on the harddrive)
- exception: N/A

OutputDevices():

- input: Hardware allowing the user to see output from the computer software, *e.g.* screen, storage
- transition: N/A
- output: Interface to allow software to communicate output to the user (*e.g.*, it provides the capability for the software to output something onto the screen or write to a file on a harddrive, or write to memory)
- exception: N/A

[It is nice if each module starts one a new page. —SS][Ok! —Author]

7 MIS of Import csv Module

7.1 Module

ImportCSV (M2)

7.2 Uses

- Data 1D Spectrum
- Sequence data structure
- Hardware-hiding

7.3 Syntax

7.3.1 Exported Access Programs

Name	In	Out	Exceptions
ReadCSV	fname: str	Spectrum	NO FILE, NOT CSV

[Having exception names with spaces in them might be a problem for your implementation. —SS][I didn't want to restrict the implementation to use exactly the same names used in this document. I would imagine other languages or other naming conventions would affect how you name your exception, so I would rather let the implementer decide the exact name to use, and put something more readable in this document. —Author]

7.4 Semantics

This module imports data from csv files and initializes a Spectrum object.

7.4.1 State Variables

N/A

7.4.2 Environment Variables

file: sequence of strings [What is the type of your environment variable? Usually a file is abstracted as a sequence of strings. —SS][Ok. —Author]

7.4.3 Access Routine Semantics

ReadCSV():
ReadCSV reads a .csv file and creates a Spectrum object with the appropriate assignations to intensity and energy range.

- input: *fname*: *str*
- transition: N/A
- output: Spectrum(data, Srange, Slabel, Sunit) [You have inputs for the constructor for Spectrum; you should identify them here. —SS][Ok, I can specify these. —Author]
- exceptions:

Exception	Condition
NO FILE	The filename does not correspond to any file in the filesystem $fname \notin filesystem$
NOT CSV	The indicated file is not a *.csv format $fname \notin \{files \| files \in .csv\}$

8 MIS of Import dm3 Module

8.1 Module

ImportDM3 (M3)

8.2 Uses

- Sequence data structure
- Hardware Hiding
- Data Spectrum Image
- Data 1D Spectrum
- Data 2D Image

8.3 Syntax

8.3.1 Exported Access Programs

Name	In	Out	Exceptions
ReadDM3	fname: string	SI | Spectrum | Image	NO FILE, NOT DM3

8.4 Semantics

This module imports data from .dm3 and initializes the appropriate data type.

8.4.1 State Variables

N/A

8.4.2 Environment Variables

- filesystem: the filesystem of the computer on which SpectrumImageAnalysisPy is being run

8.4.3 Access Routine Semantics

ReadDM3():

- input: *fname*: *str*
- transition: N/A

- output: Spectrum Image(data, dispersion, [index, value], Slabel, Sunit, Imcal, metadata) or Spectrum((data, dispersion, [index, value], Slabel, Sunit)) or Image(data, Imcal, metadata)
- exception:

Exception	**Condition**
NO FILE	The filename does not correspond to any file in the filesystem $fname \notin filesystem$
NOT DM3	The indicated file is not a *.dm3 format

9 MIS of Import h5 Module

9.1 Module

ImportH5 (M4)

9.2 Uses

- Sequence data structure
- Hardware Hiding
- Data Spectrum Image
- Data 1D Spectrum
- Data 2D Image

9.3 Syntax

9.3.1 Exported Access Programs

Name	In	Out	Exceptions
ReadH5	fname: string	SI | Spectrum | Image	NO FILE, NOT H5

9.4 Semantics

This module handles the reading of .h5 files and assignation of the data contained therein to the appropriate data type.

9.4.1 State Variables

N/A

9.4.2 Access Routine Semantics

ReadH5():

- input: *fname*: *str*
- transition: N/A
- output: Spectrum Image(data, dispersion, [index, value], Slabel, Sunit, Imcal, metadata) or Spectrum((data, dispersion, [index, value], Slabel, Sunit)) or Image(data, Imcal, metadata)

- exception:

Exception	**Condition**
NO FILE	The filename does not correspond to any file in the filesystem $fname \notin filesystem \Rightarrow$ NO_FILE
NOT H5	The indicated file is not a *.h5 format

10 MIS of Import rpl Module

10.1 Module

ImportRPL (M5)

10.2 Uses

- Sequence data structure
- Hardware Hiding
- Data Spectrum Image
- Data 1D Spectrum
- Data 2D Image

10.3 Syntax

10.3.1 Exported Access Programs

Name	In	Out	Exceptions
ReadRPL	fname: string	SI | Spectrum | Image	NO FILE, NOT RPL

10.4 Semantics

This module handles the reading of .rpl files and assigns the data contained within to the appropriate data type.

10.4.1 State Variables

N/A

10.4.2 Access Routine Semantics

ReadRPL():

- input: *fname*: *str*
- transition: N/A
- output: Spectrum Image(data, dispersion, [index, value], Slabel, Sunit, Imcal, metadata) or Spectrum((data, dispersion, [index, value], Slabel, Sunit)) or Image(data, Imcal, metadata)

- exception:

Exception	**Condition**
NO FILE	The filename does not correspond to any file in the filesystem $fname \notin filesystem \Rightarrow$ NO_FILE
NOT RPL	The indicated file is not a *.rpl format

11 MIS of Export csv Module

11.1 Module

ExportCSV (M6)

11.2 Uses

- Data Extraction 1D Slice
- Data Extraction 3D Mask
- Data 1D Spectrum
- Display 1D Spectrum
- Hardware Hiding

11.3 Syntax

11.3.1 Exported Access Programs

Name	In	Out	Exceptions
WriteCSV	Spectrum	file	FILE EXISTS

11.4 Semantics

This module writes Spectrum data to a csv file.

11.4.1 State Variables

N/A

11.4.2 Environment Variables

- filesystem

11.4.3 Access Routine Semantics

WriteCSV():

- input: Spectrum
- transition: N/A
- output: csv file containing spectrum data, written to filesystem
- exception:

Exception	**Condition**
FILE EXISTS	The filename already exists in the filesystem $fname \in filesystem \Rightarrow$ FILE_EXISTS

12 MIS of Export h5 Module

12.1 Module

ExportH5 (M7)

12.2 Uses

- Data 1D Spectrum
- Data 2D Image
- Data 3D Spectrum Image
- Data Extraction 1D Slice
- Data Extraction 2D Mask
- Data Extraction 3D Mask
- Display 1D Spectrum
- Display 2D Image
- Hardware Hiding

12.3 Syntax

12.3.1 Exported Access Programs

Name	In	Out	Exceptions
WriteH5	Spectrum | Image | Spectrum Image	file	FILE EXISTS

12.4 Semantics

This module writes Spectrum data, Image data, or Spectrum Image data to an h5 file.

12.4.1 State Variables

N/A

12.4.2 Environment Variables

- filesystem

12.4.3 Access Routine Semantics

WriteH5():

- input: Spectrum | Image | Spectrum Image
- transition: N/A
- output: h5 file containing spectrum data, image data, or spectrum image data (including any metadata), written to filesystem
- exception:

Exception	Condition
FILE EXISTS	The filename already exists in the filesystem $fname \in filesystem \Rightarrow$ FILE_EXISTS

13 MIS of Export png Module

13.1 Module

ExportPNG (M8)

13.2 Uses

- Data Extraction 2D Mask
- Data Extraction 3D Mask
- Data 2D Image
- Display 2D Image
- Hardware Hiding

13.3 Syntax

13.3.1 Exported Access Programs

Name	In	Out	Exceptions
WritePNG	Image	file	FILE EXISTS

13.4 Semantics

This module writes Image data to a png file.

13.4.1 State Variables

N/A

13.4.2 Environment Variables

- filesystem

13.4.3 Access Routine Semantics

WritePNG():

- input: Image
- transition: N/A
- output: png file containing image data, written to filesystem
- exception:

Exception	**Condition**
FILE EXISTS	The filename already exists in the filesystem $fname \in filesystem \Rightarrow$ FILE_EXISTS

14 MIS of Export rpl Module

14.1 Module

ExportRPL (M9)

14.2 Uses

- Data 1D Spectrum
- Data 2D Image
- Data 3D Spectrum Image
- Hardware Hiding

14.3 Syntax

14.3.1 Exported Access Programs

Name	In	Out	Exceptions
WriteRPL	Spectrum | Image | Spectrum Image	file	FILE EXISTS

14.4 Semantics

This module writes Spectrum data, Image data, or Spectrum Image data to an rpl file.

14.4.1 State Variables

N/A

14.4.2 Environment Variables

- filesystem

14.4.3 Access Routine Semantics

WriteRPL():

- input: Spectrum | Image | Spectrum Image
- transition: N/A
- output: rpl file containing spectrum data, image data, or spectrum image data (including any metadata), written to filesystem

- exception:

Exception	**Condition**
FILE EXISTS	The filename already exists in the filesystem $fname \in filesystem \Rightarrow$ FILE_EXISTS

15 MIS of Data Processing Richardson-Lucy Deconvolution Module

15.1 Module

RLDeconvolution (M10)

15.2 Uses

- Sequence Data Structure
- Hardware Hiding
- Data 3D Spectrum Image
- Data 1D Spectrum

15.3 Syntax

15.3.1 Exported Access Programs

Name	In	Out	Exceptions
RLDeconvolution	Spectrum, PSF, iterations	deconvolved Spectrum	DIVIDE BY ZERO
SIDeconvolution	Spectrum Image, PSF, iterations, threads	deconvolved Spectrum Image	-

15.4 Semantics

This module performs the Richardson-Lucy deconvolution algorithm on either a Spectrum or Spectrum Image, following IM2 in the SRS. [Referencing the SRS is a good idea. —SS][Thank you :) —Author]

15.4.1 State Variables

N/A

15.4.2 Environment Variables

- threads: $\mathbb{N}$, number of processing threads to use during Spectrum Image deconvolution

15.4.3 Access Routine Semantics

RLDeconvolution():

- input:
  - Spectrum, the spectrum to be deconvolved
  - Point Spread Function: Spectrum, the point spread function to deconvolve from Spectrum
  - iterations: $\mathbb{N}$, the number of iterations of the algorithm to perform
- transition: N/A
- output: Deconvolved Spectrum
- exception:

Exception	Condition
DIVIDE BY ZERO	The algorithm encounters a 0 in the denominator $\sum_l I_{PSF}(E) I^c_{real(l)}(E) = 0 \Rightarrow$ DIVIDE_BY_ZERO

SIDeconvolution():

- input:
  - Spectrum Image, the spectrum image to be deconvolved
  - Point Spread Function: Spectrum, the point spread function to deconvolve from Spectrum Image
  - iterations: $\mathbb{N}$, the number of iterations of the algorithm to perform
  - threads: $\mathbb{N}$, the number of processing threads to use in deconvolving the Spectrum Image
- transition: N/A
- output: Deconvolved Spectrum Image
- exception: N/A

16 MIS of Data Processing Normalization Module

16.1 Module

Normalization (M11)

16.2 Uses

- Sequence Data Structure
- Data 3D Spectrum Image
- Data 1D Spectrum
- Data Extraction 1D Slice

16.3 Syntax

16.3.1 Exported Access Programs

Name	In	Out	Exceptions
Normalization	Spectrum | Spectrum Image, slice	Normalized Spectrum	DIVIDE BY ZERO

16.4 Semantics

This module normalizes either a Spectrum or Spectrum Image to the sum over the range defined by the user, following IM1 in the SRS.

16.4.1 State Variables

N/A

16.4.2 Access Routine Semantics

Normalization():

- input:
  - Spectrum, the spectrum to be normalized, or Spectrum Image, the spectrum image to be normalized
  - slice
- transition: N/A
- output: Normalized Spectrum or Spectrum Image

- exception:

Exception	**Condition**
DIVIDE BY ZERO	The algorithm encounters a 0 in the denominator $\sum_{E(k=k_1)}^{E(k=k_2)} I(E(k)) = 0 \Rightarrow$ DIVIDE_BY_ZERO

17 MIS of Data Processing Gain Correction Module

17.1 Module

GainCorr (M12)

17.2 Uses

- Sequence Data Structure
- Data 3D Spectrum Image
- Data 1D Spectrum

17.3 Syntax

17.3.1 Exported Access Programs

Name	In	Out	Exceptions
GainCorrection	data, Gain Reference	Gain Corrected Spectrum	DIVIDE BY ZERO, SIZE MISMATCH

17.4 Semantics

This module corrects either a Spectrum or Spectrum Image using a gain reference (obtained from the acquisition camera hardware), following IM4 in the SRS.

17.4.1 State Variables

N/A

17.4.2 Access Routine Semantics

GainCorrection():

- input:
  - data: Spectrum, the spectrum to be corrected, or Spectrum Image, the spectrum image to be corrected
  - Gain Reference: Spectrum, a reference Spectrum obtained from the hardware
- transition: N/A
- output: Gain-corrected Spectrum or Spectrum Image
- exception:

Exception	**Condition**
DIVIDE BY ZERO	The algorithm encounters a 0 in the denominator $g(E) = 0 \Rightarrow$ DIVIDE_BY_ZERO
SIZE MISMATCH	The size of the gain correction is different from the size of the input data spectral range $len(g(E)) \neq len(data.Srange) = 0 \Rightarrow$ SIZE_MISMATCH

18 MIS of Data Processing Background Correction Module

18.1 Module

BkgndCorr (M13)

18.2 Uses

- Sequence Data Structure
- Data 3D Spectrum Image
- Data 1D Spectrum

18.3 Syntax

18.3.1 Exported Access Programs

Name	In	Out	Exceptions
BackgroundCorrection	data, Background Reference	Background Corrected Spectrum	SIZE MISMATCH

18.4 Semantics

This module corrects the background noise for either a Spectrum or Spectrum Image, following IM3 in the SRS.

18.4.1 State Variables

N/A

18.4.2 Access Routine Semantics

BackgroundCorrection():

- input:
  - data: Spectrum, the spectrum to be corrected, or Spectrum Image, the spectrum image to be corrected
  - Background Reference: Spectrum, a reference Spectrum representing the background noise in the camera
- transition: N/A
- output: Background-corrected Spectrum or Spectrum Image

- exception:

Exception	**Condition**
SIZE MISMATCH	The size of the background correction is different from the size of the input data spectral range $len(b(E)) \neq len(data.Srange) = 0 \Rightarrow$ SIZE_MISMATCH

19 MIS of Data Extraction 1D Slice Module

19.1 Module

Slice1D (M14)

19.2 Uses

- Data 1D Spectrum
- Display 1D Spectrum

19.3 Syntax

19.3.1 Exported Access Programs

Name	In	Out	Exceptions
CreateSlice	data, $[k_1, k_2]$	slice	RANGE OUTSIDE BOUNDS
IntegrateSlice	slice	integral	-

[What is the type of data, or is data a type? It looks like data is of type Spectrum, so the Input types should be Spectrum. If you would like to show the variable name, you could use the notation data: Spectrum. —SS][I found this was making the table very complicated and messy, so I left the type information in the definition. I felt that putting it in both places was repetitive, so I chose to clean up the table a bit. —Author]

19.4 Semantics

This module allows the user to extract slices from a 1D dataset for further analysis with other modules.

19.4.1 State Variables

- slice: interval of Spectrum between $[k_1, k_2]$

19.4.2 Access Routine Semantics

CreateSlice():

- input:
  - $[k_1, k_2] \in \mathbb{R}^2 | [max(min(k_1, k_2), min(data.Srange)..min(max(k_1, k_2), max(data.Srange)))] \in data.Srange$

  - data: Spectrum

- transition: Creation of slice object, using the range and the data input by the user (the slice object contains the data within this range)
- output: slice
- exception:

Exception	**Condition**
RANGE OUTSIDE BOUNDS	The user tried to select a range of values which was wholly outside the data's spectral range $[max(min(k_1, k_2), min(data.Srange)..min(max(k_1, k_2), max(data.Srange)))]$ $data.Srange \Rightarrow$ RANGE_OUTSIDE_BOUNDS

IntegrateSlice():

- input
  - slice
- transition: N/A
- output: integral over slice, $\sum_{k_1}^{k_2} data.data$
- exception: N/A

20 MIS of Data Extraction 2D Mask Module

20.1 Module

Mask2D (M15)

20.2 Uses

- Data 2D Image
- Display 2D Image

20.3 Syntax

20.3.1 Exported Access Programs

Name	In	Out	Exceptions
CreateMask	vertex list, size(data)	mask2D	RANGE OUTSIDE BOUNDS
ApplyMask	data	masked data	SIZE MISMATCH
ModifyMask	vertex list	mask2D	RANGE OUTSIDE BOUNDS

20.4 Semantics

This module allows the user to extract portions of a 2D dataset for further analysis with other modules.

20.4.1 State Variables

- mask2D, $\forall\ value \in mask, value \in \{True, False\}$, mask of the desired size, consisting of boolean values representing the desired mask from the user

20.4.2 Access Routine Semantics

CreateMask():

- input
  - vertex list: $[x_i, y_i], x_i, y_i \in \mathbb{N}$, a list of (x, y) ordered pairs of indices to access a sequence of size(data)
  - size(data): $\mathbb{N}^2$, the size of the data to which the mask and vertex list refers

- transition: Creates a mask2D object, given a list of vertices (to define the shape to mask on a 2D image) and the size of the data to be masked (to define the size of the mask).
- output: mask2D
- exception:

Exception	**Condition**
RANGE OUTSIDE BOUNDS	The user tried to select a mask with a vertex outside the data boundaries $[x_i, y_i] \notin [0..size(data_x), 0..size(data_y)] \Rightarrow$ RANGE_OUTSIDE_BOUNDS

ApplyMask():

- input
  - data: Image
- transition:
- output: masked data: Image
- exception:

Exception	**Condition**
SIZE MISMATCH	The size of the data is not the same as the size of the mask $size(mask) \neq size(data) \Rightarrow$ SIZE_MISMATCH

ModifyMask():

- input
  - vertex list: $[x_i, y_i], x_i, y_i \in \mathbb{N}$, a list of (x, y) ordered pairs of indices to access a sequence of size(data)
- transition: Update mask to the new set of input vertices
- output: mask2D
- exception:

Exception	**Condition**
RANGE OUTSIDE BOUNDS	The user tried to select a mask with a vertex outside the data boundaries $[x_i, y_i] \notin [0..size(data_x), 0..size(data_y)] \Rightarrow$ RANGE_OUTSIDE_BOUNDS

21 MIS of Data Extraction 3D Mask Module

21.1 Module

Mask3D (M16)

21.2 Uses

- Data 3D Spectrum Image
- Display 3D Spectrum Image

21.3 Syntax

21.3.1 Exported Access Programs

Name	In	Out	Exceptions
CreateMask3D	vertex list, size(data)	mask3d	RANGE OUTSIDE BOUNDS
ExtrudeMask2D	mask2D, size(data)	mask3d	SIZE MISMATCH
ExtrudeMask1D	slice, size(data)	mask3d	SIZE MISMATCH
ApplyMask3D	data	masked data	SIZE MISMATCH
ModifyMask3D	vertex list	mask3d	RANGE OUTSIDE BOUNDS

21.4 Semantics

This module allows the user to extract portions of a 3D dataset for further analysis with other modules.

21.4.1 State Variables

- mask3D, $\forall\ value \in mask, value \in \{True, False\}, size(mask3D) = size(data)$, mask of boolean values representing the desired mask from the user

21.4.2 Access Routine Semantics

CreateMask3D():

- input
  - vertex list: $[x_i, y_i, k_i], x_i, y_i, k_i \in \mathbb{N}$, a list of (x, y, k) ordered pairs of indices to access a sequence of size(data)
  - size(data): $\mathbb{N}^3$, the size of the data to which the mask and vertex list refers

- transition: Creates mask3D object, given the list of vertices from the user, representing which the indices in the data to be masked, and the size of the data, to make the mask3D the same size.
- output: mask2D
- exception:

Exception	Condition
RANGE OUTSIDE BOUNDS	The user tried to select a mask with a vertex outside the data boundaries $[x_i, y_i, k_i] \notin [0..size(data_x), 0..size(data_y), 0..size(data_k)] \Rightarrow$ RANGE_OUTSIDE_BOUNDS

ExtrudeMask2D():

- input
  - mask2D: mask2D
  - size(data): $\mathbb{N}^3$, the size of the data to which the mask and vertex list refers
- transition: Creates mask3D from mask2D by extruding it along the third dimension
- output: mask3D
- exception:

Exception	Condition
SIZE MISMATCH	The size of the data is not the same as the size of the mask $size(mask) \neq size(data) \Rightarrow$ SIZE_MISMATCH

ExtrudeMask1D():

- input
  - slice: slice
  - size(data): $\mathbb{N}^3$, the size of the data to which the mask and vertex list refers
- transition: Creates mask3D from slice by extruding it along the two extra dimensions
- output: mask3D

- exception:

Exception	Condition
SIZE MISMATCH	The size of the data is not the same as the size of the mask $size(mask) \neq size(data) \Rightarrow$ SIZE_MISMATCH

ApplyMask3D():

- input
  - data: Spectrum Image
- transition:
- output: Masked data, Spectrum Image
- exception:

Exception	Condition
SIZE MISMATCH	The size of the data is not the same as the size of the mask $size(mask) \neq size(data) \Rightarrow$ SIZE_MISMATCH

ModifyMask3D():

- input
  - vertex list: $[x_i, y_i, k_i], x_i, y_i, k_i \in \mathbb{N}$, a list of (x, y, k) ordered pairs of indices to access a sequence of size(data)
  - size(data): $\mathbb{N}^3$, the size of the data to which the mask and vertex list refers
- transition: Creates mask
- output: mask3D
- exception:

Exception	Condition
RANGE OUTSIDE BOUNDS	The user tried to select a mask with a vertex outside the data boundaries $[x_i, y_i, k_i] \notin [0..size(data_x), 0..size(data_y), 0..size(data_k)] \Rightarrow$ RANGE_OUTSIDE_BOUNDS

22 MIS of Display 1D Spectrum Module

22.1 Module

Disp1D (M17)

22.2 Uses

- Data 1D Spectrum
- Plotting library

22.3 Syntax

22.3.1 Exported Access Programs

Name	In	Out	Exceptions
plot	data	axis1D, event handler	-

22.4 Semantics

This module plots 1D spectrum data and allows event handling (*eg*, to slice the spectrum).

22.4.1 State Variables

- Spectrum axis: axis1D containing the plotted data and event handler

22.4.2 Access Routine Semantics

plot():

- input:
  - data: Spectrum
- transition: create axis and plot data on axis, initialize event handler for axis
- output:
  - axis1D: axis1D
  - event handler: axis1D
- exception: N/A

23 MIS of Display 2D Image Module

23.1 Module

Disp2D (M18)

23.2 Uses

- Data 1D Spectrum
- Plotting library

23.3 Syntax

23.3.1 Exported Access Programs

Name	In	Out	Exceptions
plot	data	axis2D, event handler	-
AddScalebar	-	-	NO CALIBRATION
ChangeContrast	[minC, maxC], data	-	OUT OF RANGE

23.4 Semantics

This module plots 2D image data and allows event handling (*eg*, to create masks on the image).

23.4.1 State Variables

- Image axis: axis2D containing the plotted data and event handler

23.4.2 Access Routine Semantics

plot():

- input:
  - data: Image
- transition: create axis and plot data on axis, initialize event handler for axis
- output:
  - axis2D: axis2D
  - event handler: axis2D

- exception: N/A

AddScalebar():

- input: N/A
- transition: Add scalebar to Image axis
- output: N/A
- exception:

Exception	**Condition**
NO CALIBRATION	No calibration exists in the image, so a scalebar cannot be added $\nexists\ data.Imcal \Rightarrow$ NO_CALIBRATION

ChangeContrast():

- input:
  - $[minC, maxC], \in \mathbb{R}^2$, the minimum and maximum contrast to stretch the colourscale to
  - data: Image
- transition: Change the contrast of the displayed image
- output: N/A
- exception:

Exception	**Condition**
OUT OF RANGE	The user tried to select a contrast range of values which was wholly outside the data's intensity limits $[max(minC, min(data.data)..maxC, max(data.data)))] \notin data.Srange \Rightarrow$ OUT_OF_RANGE

24 MIS of Display 3D Spectrum Image Module

24.1 Module

Disp3D (M19)

24.2 Uses

- Data 3D Spectrum Image
- Plotting library
- 2D image plot
- 1D spectrum plot

24.3 Syntax

24.3.1 Exported Access Programs

Name	In	Out	Exceptions
plot	data	figure3d, event handler	-

24.4 Semantics

This module arranges the elements of the 1D and 2D display modules to suit a 3D dataset, and allows connection of the different axes (through event handling) such that within one figure, all axes refer to the same dataset.

24.4.1 State Variables

- figure3d: figure containing image plot, spectrum plot, mask plot, colourbar, and image contrast histogram
- image plot: axis2D for plotting images extracted from slicing the spectrum axis
- spectrum plot: axis1D for plotting spectra extracted from masks on the image axis
- mask plot: axis2D
- colourbar axis: colourbar axis for image plot
- image contrast axis: histogram axis for image plot

24.4.2 Access Routine Semantics

plot():

- input:
  - data: Image
- transition: create axis and plot data on axis, initialize event handler for axis
- output:
  - figure3D: contains axis1D, axis2D
  - event handler for axis1D
  - event handler for axis2D
- exception: N/A

25 MIS of Data 1D Spectrum Module

25.1 Template Module

Spectrum (M20)

25.2 Uses

- Sequence Data Structure

25.3 Syntax

25.3.1 Types

Spectrum

25.3.2 Exported Access Programs

Name	In	Out	Exceptions
init	data, (Srange | (dispersion & [index, value])), Slabel, Sunit	-	WRONG DATA TYPE, LENGTH MISMATCH

25.4 Semantics

This module contains the abstract data type Spectrum, including the following state variables.

25.4.1 State Variables

- *SRange*: $\mathbb{R}^K$
- *data*: $\mathbb{R}^K$
- *index*: $\mathbb{Z}$
- *value*: $\mathbb{R}$
- *dispersion*: $\mathbb{R}$
- *Slabel*: *str*
- *Sunit*: *str*
- *metadata*: *dict*

25.4.2 Access Routine Semantics

init(): init initializes a Spectrum object.

- input:
  - *data*: intensity values, $\in \mathbb{R}^K$
  - *Srange*: spectral axis values, $\in \mathbb{R}^K$
  - *dispersion*: difference between neighbouring channels along the spectral axis, $\mathbb{R}$
  - *index*: location on the spectral axis at which *value* is, $\mathbb{Z}$
  - *value*: value of the spectral axis (in spectral axis units) at the location given by *index*, $\mathbb{R}$
  - *Slabel*: spectrum label, the name for the spectral axis (*e.g.* Energy, Wavelength), *str*
  - *Sunit*: spectrum units, the units which the spectral axis uses (*e.g.* eV, nm), *str*
- transition: Creates all state variables
- output: Spectrum [You actually have an output for a constructor. If you look at slide 19 of the MISContinued slides, you'll see an example of an ADT. You output the object here and then it can be an input for another access program. —SS][Somehow I didn't catch that, going through the slides. I'll add that for the ADTs. —Author]
- exception:

Exception	Condition
WRONG DATA TYPE	Any of the input data are the wrong type
LENGTH MISMATCH	The length of Srange is not the same as the length of data $len(Srange) \neq len(data)$

26 MIS of Data 2D Image Module

26.1 Template Module

Image (M21)

26.2 Uses

- Sequence Data Structure

26.3 Syntax

26.3.1 Types

Image

26.3.2 Exported Access Programs

Name	In	Out	Exceptions
init	data, Imcal, metadata	-	WRONG DATA TYPE, WRONG DIMENSIONS

26.4 Semantics

This module contains the abstract data type Spectrum, including the following state variables.

26.4.1 State Variables

- data: $\mathbb{R}^{X \times Y}$
- Imcal: $\mathbb{R}$
- metadata: dict

26.4.2 Access Routine Semantics

init(): init initializes an Image object.

- input:
  - *data*: intensity values, $\in \mathbb{R}^{X \times Y}$
  - *Imcal*: image calibration values (*e.g.* number of nm per pixel), $\in \mathbb{R}$

  - *metadata*: dictionary containing extra information about the source of the image (*e.g.* experimental parameters)

- transition: Creates all state variables
- output: Image
- exception:

Exception	**Condition**
WRONG_DATA_TYPE	The input data are not real numbers or the Imcal value is not a real float $data \notin \mathbb{R}^{X \times Y} \vert Imcal \notin \mathbb{R} \Rightarrow$ WRONG_DATA_TYPE
WRONG_DIMENSIONS	The input data is not 2D $size(data) \notin \mathbb{N}^2 \Rightarrow$ WRONG_DIMENSIONS

27 MIS of Data 3D Spectrum Image Module

27.1 Template Module

SI (M22)

27.2 Uses

- Sequence Data Structure

27.2.1 Types

Spectrum Image

27.2.2 Exported Access Programs

Name	In	Out	Exceptions
init	data, Srange | dispersion & [index, value], Slabel, Sunit, Imcal, metadata	-	WRONG DATA TYPE, WRONG DIMENSIONS

27.3 Semantics

This module holds spectrum image data (a 3D dataset) and associated calibrations and other related information.

27.3.1 State Variables

- *data*: $\mathbb{R}^{X \times Y \times K}$
- *Imcal*: $\mathbb{R}$
- *dispersion*: $\mathbb{R}$
- *Srange*: $\mathbb{R}^{K}$
- *index*: $\mathbb{Z}$
- *value*: $\mathbb{R}$
- *Slabel*: string
- *Sunit*: string
- *metadata*: dict

27.3.2 Access Routine Semantics

init

- input:
  - *data*: intensity values, $\in \mathbb{R}^{X \times Y \times K}$
  - *Srange*: spectral axis values, $\in \mathbb{R}^{K}$
  - *dispersion*: difference between neighbouring channels along the spectral axis, $\mathbb{R}$
  - *index*: location on the spectral axis at which *value* is, $\mathbb{Z}$
  - *value*: value of the spectral axis (in spectral axis units) at the location given by *index*, $\mathbb{R}$
  - *Slabel*: spectrum label, the name for the spectral axis (*e.g.* Energy, Wavelength), *str*
  - *Sunit*: spectrum units, the units which the spectral axis uses (*e.g.* eV, nm), *str*
  - *Imcal*: image calibration values (*e.g.* number of nm per pixel), $\in \mathbb{R}$
  - *metadata*: dictionary containing extra information about the source of the image (*e.g.* experimental parameters)
- transition: Initialize all state variables
- output: Spectrum Image
- exception:

Exception	Condition
WRONG DATA TYPE	Any of the input data are the wrong type $(data \notin \mathbb{R}^{X \times Y \times K}) \vert (Srange \notin \mathbb{R}^{K}) \vert (dispersion \notin \mathbb{R}) \vert (index \notin \mathbb{Z}) \vert (value \notin \mathbb{R}) \vert (Slabel \notin str) \vert (Sunit \notin str) \vert (Imcal \notin \mathbb{R}) \Rightarrow$ WRONG_DATA_TYPE
LENGTH MISMATCH	The length of Srange is not the same as the length of data's spectral axis $len(Srange) \neq size(data)[2] \Rightarrow$ LENGTH_MISMATCH
WRONG DATA TYPE	The input data are not real numbers or the Imcal value is not a real float $data \notin \mathbb{R}^{X \times Y} \vert Imcal \notin \mathbb{R} \Rightarrow$ WRONG_DATA_TYPE
WRONG DIMENSIONS	The input data is not 2D $size(data) \notin \mathbb{N}^2 \Rightarrow$ WRONG_DIMENSIONS

28 MIS of Sequence Data Structure Module

28.1 Template Module

Sequence (M23)

28.2 Uses

N/A

28.3 Syntax

28.3.1 Type

- Sequence

28.3.2 Exported Access Programs

Name	In	Out	Exceptions
CreateSeq	data	Sequence	-
ModifySeq	Sequence	Sequence	-

28.4 Semantics

This module holds the sequence structure and functions for performing various calculations on sequences.

28.4.1 State Variables

- sequence: $\mathbb{C}^N, dim(N) \in \mathbb{N}$

28.4.2 Access Routine Semantics

CreateSeq():

- input: data, $\mathbb{C}^N$
- transition: Create sequence variable
- output: sequence, $\mathbb{C}^N$
- exception: N/A

ModifySeq():

- input: sequence, $\mathbb{C}^N$

- transition: Modify sequence by some operation, including but not limited to, addition, subtraction, multiplication, division, *etc.*
- output: sequence, $\mathbb{C}^N$
- exception: N/A

29 MIS of Plotting Library Module

29.1 Module

Plotting (M24)

29.2 Uses

- Hardware Hiding Module

29.3 Syntax

29.3.1 Exported Access Programs

Name	In	Out	Exceptions
plot	data	window	-

29.4 Semantics

This module is the basis for plotting 1D and 2D data and handling events such as mouse clicks and keyboard presses.

29.4.1 State Variables

- figure
- axis1D
- axis2D
- event handler

29.4.2 Environment Variables

- window: 2D on-screen display of plot figure

29.4.3 Access Routine Semantics

plot():

- input: data, $\mathbb{R}^K | \mathbb{R}^{X \times Y}$
- transition: Creates a figure to display the input data, with a 1D plot axis for 1D data or a 2D plot axis for 2D data. Provides handling for events such as mouse clicks or keyboard key presses and options to format the display.
- output: window

- exception:

[Great work. There is too much in here for me to go over everything, but I am confident that you are on the right track with your documentation. Hopefully maintaining the documentation with the code will not be too difficult. I look forward to your feedback in the future to see how easy/hard it is to keep the documentation in sync. —SS]

References

[1] D. M. Hoffman and P. A. Strooper, *Software Design, Automated Testing, and Maintenance: A Practical Approach*. New York, NY, USA: International Thomson Computer Press, 1995.

[2] C. Ghezzi, M. Jazayeri, and D. Mandrioli, *Fundamentals of Software Engineering*. Upper Saddle River, NJ, USA: Prentice Hall, 2nd ed., 2003.