SpectrumImageAnalysisPy

Isobel Bicket

October 24, 2017

1 Revision History

Date	Version	Notes
October 24, 2017	1.0	Initial draft

2 Symbols, Abbreviations and Acronyms

symbol	description
T	Test

[symbols, abbreviations or acronyms – you can reference the SRS tables if needed —SS]

Contents

List of Tables

List of Figures

This document outlines the different tests and assessment tools which will be used for assessing the performance of SpectrumImageAnalysisPy. The testing will be done to build confidence in the performance of the program and allow the end user to use it with more assurity that it is behaving correctly and any artifacts seen in the data are either a known result of the program, or independent of the use of SpectrumImageAnalysisPy to process the data.

3 General Information

The testing outlined in this document gives valuable contributions to the level of trust in the performance of the software. Not only will testing the software build confidence that it is performing correctly, but the results of the tests will also tell the user what to expect when running the software, for instance in terms of performance milestones. Testing not only ensures that the software will, within predictive power, fulfill its functional requirements, but gives the progress and effectiveness of the software in fulfulling the non-functional requirements.

3.1 Purpose

The purpose of this document is to describe the tests which will be carried out on SpectrumImageAnalysisPy to verify that it meets the requirements described in the SRS.

3.2 Scope

This document will cover the verification plan for testing the software SpectrumImageAnalysisPy, including the system tests towards verifying that the software meets the functional and non-functional requirements; and the unit tests, of which a non-exhaustive list is given. The document does not cover the validation of the software algorithms.

3.3 Overview of Document

It begins by outlining the structure of the document, followed by an overview of the testing arrangements. Specific details follow on the tests which will be performed to test whether or not SpectrumImageAnalysisPy satisfies the

functional and non-functional requirements, as well as details of the unit tests which will be written. The unit tests themselves may satisfy some of the functional requirements without needing system testing, these will be detailed under Unit Testing Plan as appropriate.

4 Plan

This section details the plan to be followed when testing the software, including those involved in the testing, the testing approach, and the verification tools which will be used.

4.1 Software Description

The software, SpectrumImageAnalysisPy, is designed as a data analysis tool for electron energy loss spectroscopy and cathodoluminescence spectrum imaging data. It provides several data processing routines to the user and a graphical user interface to navigate the 3D data set and export desired spectra or image slices. The requirements and expectations for SpectrumImageAnalysisPy are detailed in the Software Requirements Specification.

4.2 Test Team

The test team has one member: Isobel Bicket.

4.3 Automated Testing Approach

Unit testing will be performed on the functions within the code, as an automated task to be done with every update of the code. The running of automated tests also provides regression testing and integration testing of new features and updates. The goal of testing is 100% code coverage.

4.4 Verification Tools

The following verification tools will be used

- Python unittest library: for performing unit tests on the code;
- Python coverage library: to determine the coverage of the tests;

- Deconvo.m: a Richardson-Lucy deconvolution algorithm implementation in Matlab, written by Dr. E.P. Bellido [1];
- Python HypersPy library: a Python-based library for EELS and EDX spectrum processing [2].

[Thoughts on what tools to use, such as the following: unit testing framework, valgrind, static analyzer, make, continuous integration, test coverage tool, etc. —SS]

4.5 Non-Testing Based Verification

The following non-testing based verification methods will be used to assess the performance of SpectrumImageAnalysisPy

- Code review: a detailed code review will be performed by an external party and feedback provided;
- User survey: a group of qualified users will be asked to process a sample dataset using the software and will be polled on their user experience.

[List any approaches like code inspection, code walkthrough, symbolic execution etc. Enter not applicable if that is the case. —SS]

5 System Test Description

5.1 Tests for Functional Requirements

5.1.1 Inputs

Spectrum Image Input

1. SI Data array

   Type: Functional, Dynamic, Manual, Static etc.

   Initial State:

   Input:

   Output:

   How test will be performed:

1. Type: Functional, Dynamic, Manual, Static etc.

   Initial State: DD1

   Input:

   Output:

   How test will be performed:

5.1.2 Area of Testing2

...

5.2 Tests for Nonfunctional Requirements

5.2.1 Area of Testing1

Title for Test

1. test-id1

   Type:

   Initial State:

   Input/Condition:

   Output/Result:

   How test will be performed:

2. test-id2

   Type: Functional, Dynamic, Manual, Static etc.

   Initial State:

   Input:

   Output:

   How test will be performed:

5.2.2 Area of Testing2

...

5.3 Traceability Between Test Cases and Requirements

6 Unit Testing Plan

6.1 Inputs

Spectrum Image Input

1. SI Data array with energy range (EELS)

   Type: Functional, Dynamic, Unit

   Initial State: None

   Input: 3D data array, present in memory; use standard calibration variabilities
   For example, the following values might be used to create an SI:

   - $x = [0, 1]$
   - $y = [0, 1]$
   - x calibration = 1 nm/pixel
   - energy range = range$(0, 10)$ eV

   Output: Spectrum Image stored within SpectrumImageAnalysisPy; the first and second axes should correspond to spatial dimensions, while the third axis should correspond to a spectral dimension.

   How test will be performed: Create a 3D data array and attempt to initialize a spectrum image in SpectrumImageAnalysisPy. No errors should be raised. The spectrum image axes should be read in the correct order (x, y, E), and the calibrations should be applied to the correct axes. Addresses Functional Requirement.

2. SI Data array with dispersion (EELS)

   Type: Functional, Dynamic, Unit

   Initial State: None

Input: 3D data array, present in memory; use standard calibration variabilities
For example, the following values might be used to create an SI:

- $x = [0, 1]$
- $y = [0, 1]$
- x calibration = 1 nm/pixel
- dispersion = 0.01 eV/pixel
- $E(0) = -2$ eV

Output: Spectrum Image stored within SpectrumImageAnalysisPy; the first and second axes should correspond to spatial dimensions, while the third axis should correspond to a spectral dimension.

How test will be performed: Create a 3D data array and attempt to initialize a spectrum image in SpectrumImageAnalysisPy. No errors should be raised. The spectrum image axes should be read in the correct order (x, y, E), and the calibrations should be applied to the correct axes.

3. SI Data array with wavelength range (CL)

   Type: Functional, Dynamic, Unit

   Initial State: None

   Input: 3D data array, present in memory; use standard calibration variabilities
   For example, the following values might be used to create an SI:

   - $x = [0, 1]$
   - $y = [0, 1]$
   - x calibration = 1 nm/pixel
   - wavelength range = range$(0, 10)$ nm

   Output: Spectrum Image stored within SpectrumImageAnalysisPy; the first and second axes should correspond to spatial dimensions, while the third axis should correspond to a spectral dimension.

   How test will be performed: Create a 3D data array and attempt to initialize a spectrum image in SpectrumImageAnalysisPy. No errors

should be raised. The spectrum image axes should be read in the correct order (x, y, E), and the calibrations should be applied to the correct axes.

4. SI .dm3 file

   Type: Functional, Dynamic, Unit

   Initial State: None

   Input: A .dm3 file, containing a fabricated spectrum image. The .dm3 file will be created using Digital Micrograph (Gatan Microscopy Suite Software) software [3] with the following parameters:

   - $x = [0, 1]$
   - $y = [0, 1]$
   - x calibration = 1 nm/pixel
   - dispersion = 0.01 eV/pixel
   - $E(0) = -2$ eV

   Output: EELS Spectrum Image stored within SpectrumImageAnalysisPy, with metadata assigned to appropriate values

   How test will be performed: Read in the .dm3 file with SpectrumImageAnalysisPy. Check that no errors are raised. Display the spectrum image to manually check that the data was read in correctly. Check spatial and spectral calibration to verify metadata was assigned correctly.

   [There is a potential problem with the .dm3 file, since there are many version of Digital Micrograph out there and many different set-ups, all of which may format their .dm3 slightly differently. Not sure on the best way to test this, without asking several other labs for pieces of data (which are not usually shared readily) and trying out all of them. —Author]

5. SI .h5 file

   Type: Functional, Dynamic, Unit.

   Initial State: None

Input: .h5 file, containing an acquired or fabricated spectrum image. The .h5 spectrum image file originates from Odemis software [4], it may be possible to fabricate a spectrum image using the scripting interface, using the following values for the calibration parameters, otherwise it may be necessary to use an acquired data file.

- $x = [0, 1]$
- $y = [0, 1]$
- x calibration = 1 nm/pixel
- wavelength range = range$(0, 10)$ nm

Output: CL Spectrum Image stored within SpectrumImageAnalysisPy, with metadata assigned to appropriate values.

How test will be performed: Read in the .h5 file with SpectrumImageAnalysisPy. Check that no errors are raised. Check spatial and spectral calibration to verify metadata was assigned correctly.

Spectrum Input

1. Spectrum Data array

   Type: Functional, Dynamic, Unit.

   Initial State:

   Input: 3D data array, present in memory. For example, the following data may be used to initialize a Spectrum:

   - energy values = ones$(size(10))$ counts
   - energy range = range$(0, 10)$ eV

   Output: Spectrum Image stored within SpectrumImageAnalysisPy; the first and second axes should correspond to spatial dimensions, while the third axis should correspond to a spectral dimension.

   How test will be performed: Create a 3D data array and attempt to read it with SpectrumImageAnalysisPy. Check spectral calibration to verify metadata was assigned correctly.

2. Spectrum .csv file

   Type: Functional, Dynamic, Unit.

   Initial State:

   Input: .csv file, containing an acquired spectrum. The following data may be used to create a spectrum csv file:

   - Intensity = ones($size(10)$) counts
   - energy range = range$(0, 10)$ eV

   Output: Spectrum Image stored within SpectrumImageAnalysisPy, with metadata assigned to appropriate values

   How test will be performed: A csv file containing spectrum data will be imported. No errors should be raised. The unit test will check that the intensity and energy range are assigned to the appropriate variables, as well as the spectral calibration.

3. Spectrum Data array - complex numbers

   Type: Functional, Dynamic, Unit.

   Initial State:

   Input: 3D data array, present in memory. For example, the following data may be used to initialize a Spectrum:

   - energy values = ones(size(10)) + $j *$ ones(size(10)) counts
   - energy range = range$(0, 10)$ eV

   Output: Spectrum Image stored within SpectrumImageAnalysisPy; the first and second axes should correspond to spatial dimensions, while the third axis should correspond to a spectral dimension.

   How test will be performed: Create a 3D data array and attempt to read it with SpectrumImageAnalysisPy. Check spectral calibration to verify metadata was assigned correctly.

References

[1] E. P. Bellido, D. Rossouw, and G. A. Botton, "Toward 10 meV Electron Energy-Loss Spectroscopy Resolution for Plasmonics," *Microscopy and Microanalysis*, vol. 20, pp. 767–778, June 2014.

[2] F. de la Peña, T. Ostasevicius, V. T. Fauske, P. Burdet, P. Jokubauskas, M. Nord, E. Prestat, M. Sarahan, K. E. MacArthur, D. N. Johnstone, J. Taillon, J. Caron, T. Furnival, A. Eljarrat, S. Mazzucco, V. Migunov, T. Aarholt, M. Walls, F. Winkler, B. Martineau, G. Donval, E. R. Hoglund, I. Alxneit, I. Hjorth, L. F. Zagonel, A. Garmannslund, C. Gohlke, I. Iyengar, and H.-W. Chang, "hyperspy/hyperspy: Hyperspy 1.3," May 2017.

[3] "Gatan Microscopy Suite Software."

[4] D. BV, "ODEMIS: Integrated software for microscopy solutions | DELMIC."

7 Appendix

This is where you can place additional information.

7.1 Symbolic Parameters

The definition of the test cases will call for SYMBOLIC_CONSTANTS. Their values are defined in this section for easy maintenance.

7.2 Usability Survey Questions?

This is a section that would be appropriate for some teams.