System Verification and Validation Plan for ImgBeamer

Joachim de Fourestier

February 17, 2023

1 Revision History

Date	Version	Notes
2023/02/17	1.0	First version
Date 2	1.1	Notes

Contents

1 Revision History **i**

2 Symbols, Abbreviations and Acronyms **iv**

3 General Information **1**
- 3.1 Summary . . . 1
- 3.2 Objectives . . . 1
- 3.3 Relevant Documentation . . . 1

4 Plan **2**
- 4.1 Verification and Validation Team . . . 2
- 4.2 SRS Verification Plan . . . 2
- 4.3 Design Verification Plan . . . 3
- 4.4 Verification and Validation Plan Verification Plan . . . 3
- 4.5 Implementation Verification Plan . . . 3
- 4.6 Automated Testing and Verification Tools . . . 4
- 4.7 Software Validation Plan . . . 4

5 System Test Description **4**
- 5.1 Tests for Functional Requirements . . . 4
  - 5.1.1 Image Import and Export . . . 5
  - 5.1.2 Spot Profile and Imaging Parameters . . . 6
  - 5.1.3 Image Quality Metric . . . 7
- 5.2 Tests for Nonfunctional Requirements . . . 8
  - 5.2.1 Area of Testing1 . . . 8
  - 5.2.2 Area of Testing2 . . . 8
- 5.3 Traceability Between Test Cases and Requirements . . . 8

6 Unit Test Description **9**
- 6.1 Unit Testing Scope . . . 9
- 6.2 Tests for Functional Requirements . . . 9
  - 6.2.1 Module 1 . . . 9
  - 6.2.2 Module 2 . . . 10
- 6.3 Tests for Nonfunctional Requirements . . . 10
  - 6.3.1 Module ? . . . 10
  - 6.3.2 Module ? . . . 11
- 6.4 Traceability Between Test Cases and Modules . . . 11

7 Appendix **12**
7.1 Symbolic Parameters 12
7.2 Usability Survey Questions? 12

List of Tables

1 Table of the VnV Team Members 2

List of Figures

[Remove this section if it isn't needed —SS]

2 Symbols, Abbreviations and Acronyms

symbol	description
GUI	Graphical User Interface
ROI	Region of Interest
SRS	Software Requirements Specification
T	Test
UI	User Interface
VnV	Verification and Validation

[symbols, abbreviations or acronyms — you can simply reference the SRS [4] tables, if appropriate —SS]

[Remove this section if it isn't needed —SS]

This document lays out the Verification and Validation (VnV) plan of the ImgBeamer (scanning electron microscope image formation) as described by the Software Requirements Specification (SRS) document [4]. Testing of the software and its components is conducted to build confidence in its usability, accuracy, and ultimately whether it meets the SRS.

3 General Information

3.1 Summary

The ImgBeamer software aims to be a demonstration and learning tool of how a scanning electron microscope (SEM) formulates an image. The idea is to help visualize the influence trends of imaging parameters (as defined in the SRS) on image quality (or clarity) and resolution.

3.2 Objectives

The objective is to produce a tool that is easy to use (*usability*). It should feel intuitive to the user and should provide easy to understand information. This is so that the user may identify any trends in how the final image is affected with relative ease and confidence. As a secondary goal, the *accuracy* in the trends are important. As such, the image metrics should be straightforward to understand and provide some kind of assurance away from the subjective nature of image quality.

3.3 Relevant Documentation

There are multiple design documents that provide the important and intimate details to understand some of the concepts that are being tested. These documents are the following:

- Problem Statement [3]
- SRS [4]
- VnV Report [5]

4 Plan

In this section, multiple plans are described to test and inspect the software with an emphasis on *usability*. Multiple approaches and perspectives will be employed by the VnV team (4.1) to help build confidence in the requirements, to avoid any missed important details, and to deliver on the qualities as mentioned in the objective (3.2).

4.1 Verification and Validation Team

The members of the VnV team as well their individual roles are listed in the following table:

Role	Name
Project Supervisor	Dr. Spencer Smith
Author	Joachim de Fourestier
Domain Expert	Karen Wang
SRS Reviewer	Jason Balaci
VnV Plan Reviewer	Sam Crawford
MG + MIS Reviewer	Lesley Wheat
Expert Consultant	Dr. Nabil Bassim
Expert Consultant	Michael W. Phaneuf

Table 1: Table of the VnV Team Members

[Dr. Bassim is my PhD Supervisor, and Mr. Phaneuf is my co-supervisor (employer/industry), since I am doing an industrial PhD. Much of this project was originally conceptualized with the help and guidance of Mr. Phaneuf. —Author]

4.2 SRS Verification Plan

The SRS document is reviewed by the project supervisor, the SRS reviewer and the author. Some input may be given by the expert consultants if time permits. Most of the feedback has been provided as issues on GitHub, or as annotated documents, or by verbal exchange. The author is then expected

make the resolve the issues and makes accordingly throughout the development of the project. The key objectives are to verify that the software requirements and the documentation are coherent and sound to the intended audience as defined in the SRS.

4.3 Design Verification Plan

Much of the conceptualization was done after having multiple discussions with the expert consultants and having done literature review for preexisting tools and documentation relevant to the project. Decision are made with focus on the usability, with little to no setup being required. The design and implementation is documented in the MG / MIS [1, 2]. The VnV team as well as any volunteer users are welcome to provide their input (through GitHub issues). Eventually, the project may be published in a journal where the software ImgBeamer and its accompanying documentation will likely be rigorously reviewed.

[Not sure what was appropriate to put here. —Author]

4.4 Verification and Validation Plan Verification Plan

The goal is to uncover any mistakes and reveal any risks through the supervision and review of the VnV team members. Once most of the work has been done, the work and accompanying documentation shall undergo a final vetting process. Mainly, the team will check whether the documented testing plans and verification process have been accomplished and the requirements fulfilled.

4.5 Implementation Verification Plan

Much of the software will be tested manually by users. This will include checking for any inconsistencies, bugs in the graphical user interface (GUI), and any unexpected artifacts in images produced. The image metrics will be tested using unit tests (section 6). For the all the code implemented, linters will be used as mentioned in section 4.6. As a control for the image metrics, they will be calculated using the same image as the ground truth reference to rule out any baseline or unexpected factors in the implementations themselves.

4.6 Automated Testing and Verification Tools

The image quality metric shall be unit-tested using pytest for automated testing of any algorithms implemented in Python and QUnit shall be used for those implemented in JavaScript. The unit tests are listed in section 6. As for linter, flake8 shall be used for Python code and ESLint for JavaScript code. The sewar python package will be used as a reference implementation in Python for the image quality metrics.

4.7 Software Validation Plan

A *usability* survey will be conducted to evaluate the user experience and whether the GUI is intuitive enough to the intended users as described in the SRS [4]. An *accuracy* survey will be conducted to assess the user-perceived image quality. The trends identified in the surveys results will be compared to the calculated image metrics. As a control for the images produced, a manual and an automated test will be conducted to verify if an image identical (or with unperceivable difference) to ground truth image can be reproduced. The compared images shall be in the *accuracy* survey for confirmation. This can be compared as well using the image metrics.

[I think I might be confusing the Software and the Implementation validation plans... —Author]

5 System Test Description

In this section, the system tests that will be conducted as described in detail. These tests will be used to verify the fulfillment of the requirements as listed in the SRS [4]. Most, if not all, of the tests listed here will be manually performed. Automatic tests will be unit tests as described in section 6.

5.1 Tests for Functional Requirements

The tests present will verify whether the functional requirements (as listed in the SRS [4]) are met and validate that the outputs are as expected.

5.1.1 Image Import and Export

To satisfy R1 from the SRS [4], an input image for the follow formats shall be accepted provided they follow the input data constraints. Some preloaded example images shall be included and be available to the user for use as well. As well as for R6, the software should allow for export of the resulting image.

- PNG
- JPG
- BMP

1. **T1**: Test import for PNG/JPG/BMP format

   Control: Manual

   Initial State: Loaded and idle.

   Input: A non-corrupt (valid) PNG, JPG, or BMP image file.

   Output: The image should be visible and be displayed as expected as the ground truth image.

   Test Case Derivation: Theses are some of the most common image file formats and should be compatible with the software.

   How test will be performed: The user will click the "Load image" button and select an image to import.

2. **T2**: Test PNG Image Export

   Control: Manual

   Initial State: Loaded and idle.

   Input: not needed, the software defaults to a preloaded image if none is given.

   Output: The output image file should look the same as what is displayed in the "Resulting Image" display.

   Test Case Derivation: [Not sure what this means, and how this differs from what's explained Output. —Author]

   How test will be performed: The user will click the "Export image" button and choose where to save the image file.

5.1.2 Spot Profile and Imaging Parameters

To satisfy R2 and R5 from the SRS [4], the software should accept input from the user to change the width, height, and rotation which define an ellipsoid shape. This shape is then used as the spot to sample the ground truth image. To satisfy R3, the software should accept user input for a real positive number for the pixel size. TO satisfy R4, the user should be able to specify a subregion (or ROI) for processing.

1. **T3**: Spot Width and Height

   Control: Manual

   Initial State: Running and idle.

   Input: Width and Height as positive real numbers relative.

   Output: The spot layout should reflect these changes display an updated arrangement with the given shape. The resulting image should exhibit expected effects as document the SRS figures [4].

   Test Case Derivation: [not needed in this case? —Author]

   How test will be performed: Either by scrolling in the visual spot shape UI or by number input in the GUI.

2. **T4**: Spot Rotation

   Control: Manual

   Initial State: Running and idle.

   Input: An angle in degrees as a real number.

   Output: The spot layout should reflect the updated arrangement with the rotated spots. The resulting image should exhibit expected effects as document the SRS figures [4].

   Test Case Derivation: [see output? —Author]

   How test will be performed: By dragging the rotation node in the visual spot shape UI.

3. **T5**: Raster Grid / Pixel Size

   Control: Manual

   Initial State: Running and idle.

Input: Positive real numbers for the rows and columns used for the raster grid to calculate the number of pixels and their size.

Output: The resulting image resolution should grow or shrink according to the pixel size given.

Test Case Derivation: [see output? —Author]

How test will be performed: The user shall input the number of rows and columns.

5.1.3 Image Quality Metric

To satisfy R7, the user be able to see a larger number for a resulting image that is closer to ground truth, and a smaller number otherwise.

1. **T6**: Higher metric value

   Control: Manual

   Initial State: Running and idle.

   Input: Spot size between 100% to 130%.

   Output: A number that is higher than if the spot size were outside the 100% to 130% range.

   Test Case Derivation: The resulting image should look somewhat sharp.

2. **T7**: Low metric value

   Control: Manual

   Initial State: Running and idle.

   Input: Spot size from 0% to 90% or from 140% or more.

   Output: A number that is lower than if the spot size were inside the range of 100% to 130%.

   Test Case Derivation: The resulting image should look somewhat blurry or overly sharp or pixelated.

5.2 Tests for Nonfunctional Requirements

[The nonfunctional requirements for accuracy will likely just reference the appropriate functional tests from above. The test cases should mention reporting the relative error for these tests. Not all projects will necessarily have nonfunctional requirements related to accuracy —SS]

[Tests related to usability could include conducting a usability test and survey. The survey will be in the Appendix. —SS]

[Static tests, review, inspections, and walkthroughs, will not follow the format for the tests given below. —SS]

5.2.1 Area of Testing1

1. test-id1

   Type: Functional, Dynamic, Manual, Static etc.

   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

[Provide a table that shows which test cases are supporting which requirements. —SS]

6 Unit Test Description

[Reference your MIS (detailed design document) and explain your overall philosophy for test case selection. —SS] [This section should not be filled in until after the MIS (detailed design document) has been completed. —SS]

6.1 Unit Testing Scope

[What modules are outside of the scope. If there are modules that are developed by someone else, then you would say here if you aren't planning on verifying them. There may also be modules that are part of your software, but have a lower priority for verification than others. If this is the case, explain your rationale for the ranking of module importance. —SS]

6.2 Tests for Functional Requirements

[Most of the verification will be through automated unit testing. If appropriate specific modules can be verified by a non-testing based technique. That can also be documented in this section. —SS]

6.2.1 Module 1

[Include a blurb here to explain why the subsections below cover the module. References to the MIS would be good. You will want tests from a black box perspective and from a white box perspective. Explain to the reader how the tests were selected. —SS]

1. test-id1

   Type: [Functional, Dynamic, Manual, Automatic, Static etc. Most will be automatic —SS]

   Initial State:

   Input:

   Output: [The expected result for the given inputs —SS]

   Test Case Derivation: [Justify the expected value given in the Output field —SS]

   How test will be performed:

2. test-id2

   Type: [Functional, Dynamic, Manual, Automatic, Static etc. Most will be automatic —SS]

   Initial State:

   Input:

   Output: [The expected result for the given inputs —SS]

   Test Case Derivation: [Justify the expected value given in the Output field —SS]

   How test will be performed:

3. ...

6.2.2 Module 2

...

6.3 Tests for Nonfunctional Requirements

[If there is a module that needs to be independently assessed for performance, those test cases can go here. In some projects, planning for nonfunctional tests of units will not be that relevant. —SS]

[These tests may involve collecting performance data from previously mentioned functional tests. —SS]

6.3.1 Module ?

1. test-id1

   Type: [Functional, Dynamic, Manual, Automatic, Static etc. Most will be automatic —SS]

   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:

6.3.2 Module ?

...

6.4 Traceability Between Test Cases and Modules

[Provide evidence that all of the modules have been considered. —SS]

References

[1] J. de Fourestier. Module guide for imgbeamer, 2023. URL https://github.com/joedf/CAS741_w23/blob/main/docs/Design/SoftArchitecture/MG.pdf.

[2] J. de Fourestier. Module interface specification for imgbeamer, 2023. URL https://github.com/joedf/CAS741_w23/blob/main/docs/Design/SoftDetailedDes/MIS.pdf.

[3] J. de Fourestier. Problem statement and goals: Imgbeamer, 2023. URL https://github.com/joedf/CAS741_w23/blob/main/docs/ProblemStatementAndGoals/ProblemStatement.pdf.

[4] J. de Fourestier. Software requirements specification for imgbeamer: Scanning electron microscope image formation, 2023. URL https://github.com/joedf/CAS741_w23/blob/main/docs/SRS/SRS.pdf.

[5] J. de Fourestier. Verification and validation report: Imgbeamer, 2023. URL https://github.com/joedf/CAS741_w23/blob/main/docs/VnVReport/VnVReport.pdf.

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 projects. —SS]