Verification and Validation Report: ImgBeamer

Joachim de Fourestier

April 4, 2023

1 Revision History

Date	Version	Notes
2023/03/26	0.1.0	Creation
2023/04/02	0.1.1	Start requirements testing sections
2023/04/03	0.2.0	Fill in the Functional Requirements Evaluation section, along with test images
2023/04/04	0.3.0	Complete NFR testing sections
	0.3.1	Start unit testing section

2 Symbols, Abbreviations and Acronyms

symbol	description
GUI	Graphical User Interface
MG	Module Guide
MIS	Module Interface Specification
NFR	Non Functional Requirement
R	Requirement
SRS	Software Requirements Specification
T	Test
VnV	Verification and Validation

See the SRS [3], VnV Plan [4], MG [1], and MIS [2] Documentation for additional items.

Contents

List of Tables

List of Figures

3 Report Purpose

This purpose of this report is to document the tasks accomplished and testing results as part the verification and validation process of ImgBeamer as laid out in the VnV Plan [4]. The code documentation along with notes on developer setup and testing is available at: https://joedf.github.io/ImgBeamer/jsdocs/index.html The software design documentation is available at: https://github.com/joedf/CAS741_w23. The source code is available at: https://github.com/joedf/ImgBeamer/tree/cas741

4 Functional Requirements Evaluation

In this section, we report the measures that were taken to evaluate whether the functional requirements (as listed in the SRS [3]) were met. Most of these tests were executed manually due to the complex nature of visual information, GUI testing, and the subjectivity of image quality.

4.1 Image Import and Export (R1 and R6)

This section is focused on testing the image import and export functionalities.

4.1.1 T1: Test import for PNG/JPG/BMP format (R1)

The software could successfully load or import valid (non-corrupt) images files in the PNG, JPG, and BMP formats. An example is shown in figure 1.

4.1.2 T2: Test PNG Image Export (R6)

The software could successfully export valid PNG image files of the resulting image (see figure 2).

4.2 Spot Profile and Imaging Parameters (R2, R3, R4, and R5)

This section focuses on testing the sampling and image rendering based on the given imaging parameters and subregion / ROI. The ground truth / input image used for these tests is depicted in figure 3 with the cyan overlay depicting the subregion area.

4.2.1 T3: Spot Width and Height - Exact-sampling (R2 and R5)

The test was passed as shown in figure 4.

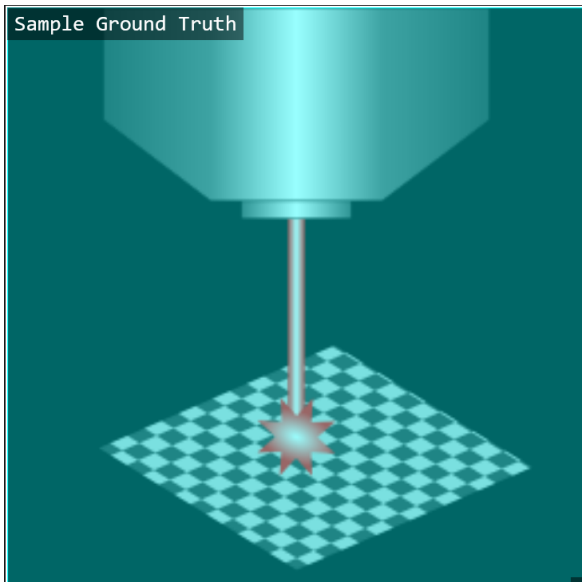

Figure 1: An example loaded test image.

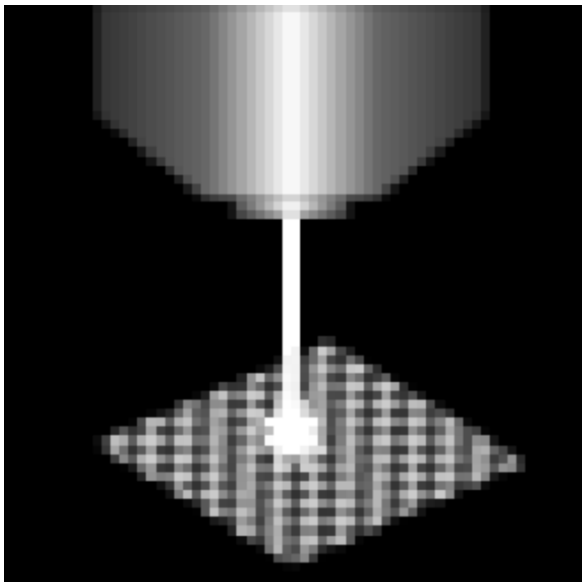

Figure 2: The image was faithfully reproduced (in grayscale with some imaging parameters change for clarity).

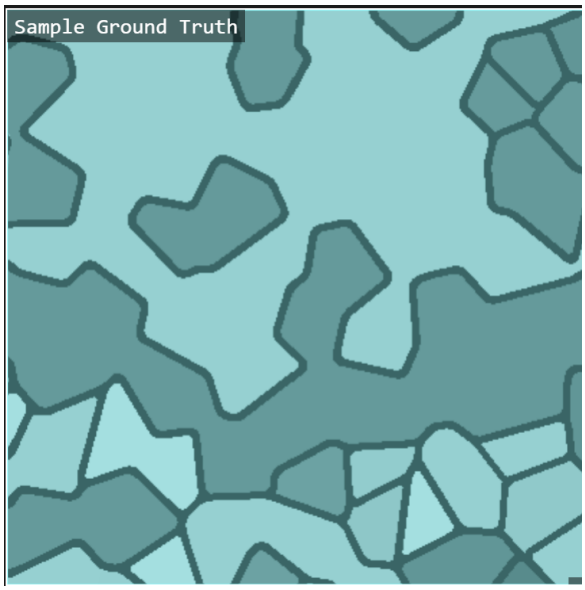

Figure 3: The ground truth image used for testing with the subregion set to cover the entire image.

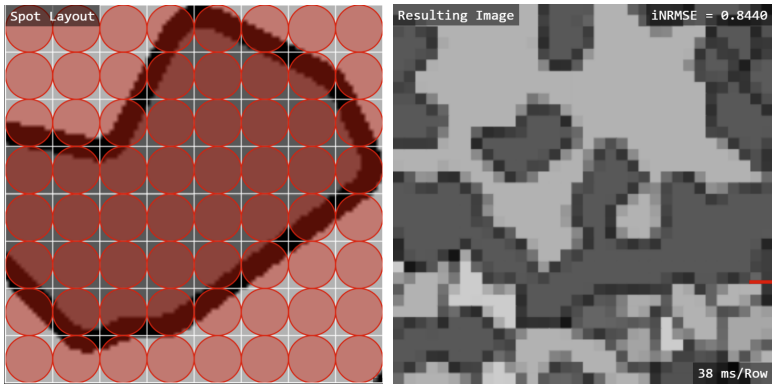

Figure 4: The spot layout and resulting image matching the expected test result: Spot size of 100 by 100%.

4.2.2 T4: Spot Width and Height - Under-sampling (R2 and R5)

The test was passed as shown in figure 5.

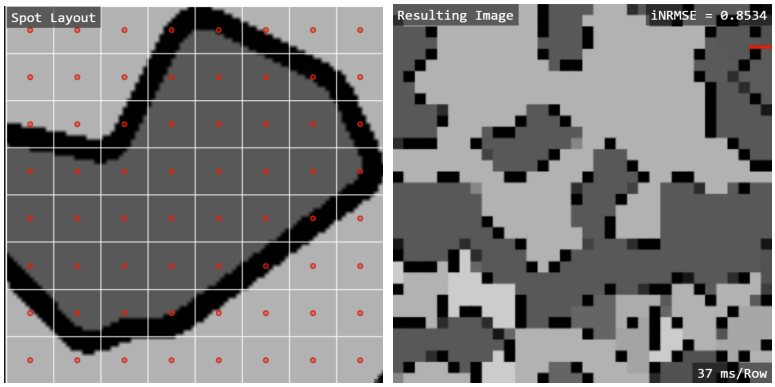

Figure 5: The spot layout and resulting image matching the expected test result: Spot size of 10 by 10%.

4.2.3 T5: Spot Width and Height - Over-sampling (R2 and R5)

The test was passed as shown in figure 6.

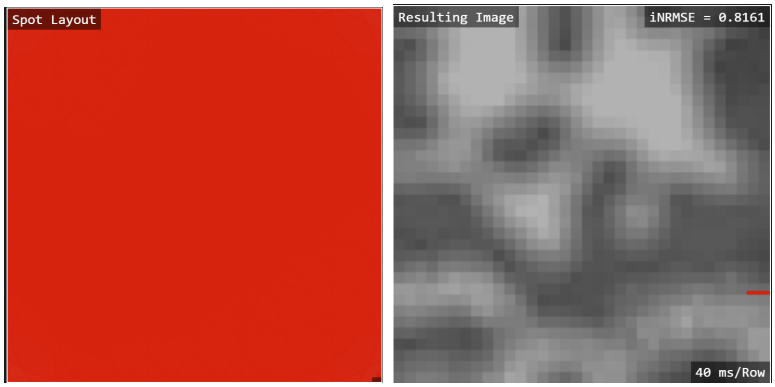

Figure 6: The spot layout and resulting image matching the expected test result: Spot size of 500 by 500%.

4.2.4 T6: Spot Rotation - Astigmatism (R2 and R5)

The test was passed as shown in figure 7.

4.2.5 T7: Raster Grid / Pixel Size (R3 and R5)

The test was passed as shown in figure 8.

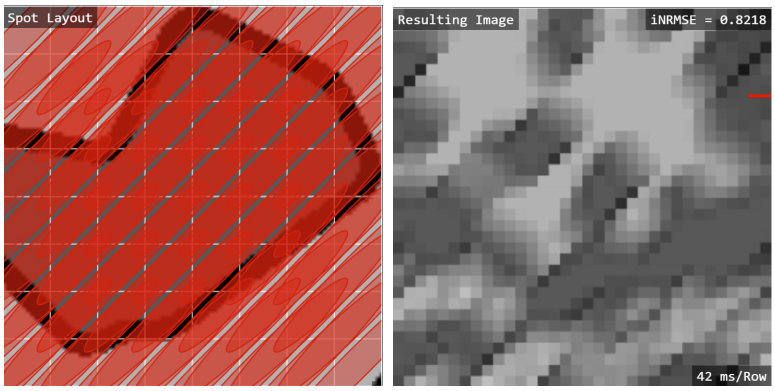

Figure 7: The spot layout and resulting image matching the expected test result: Spot size of 60 by 500% at 45 degrees rotation.

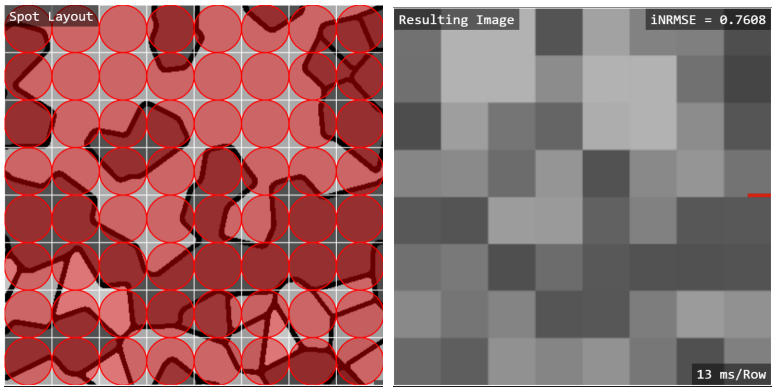

Figure 8: The spot layout and resulting image matching the expected test result: an 8 by 8 pixel image.

4.2.6 T8: Subregion / ROI (R4 and R5)

The test was passed as shown in figure 9.

4.2.7 T9: Ground Truth Reproduction (R1, R2, R3, and R6)

The test was passed as shown in figure 10.

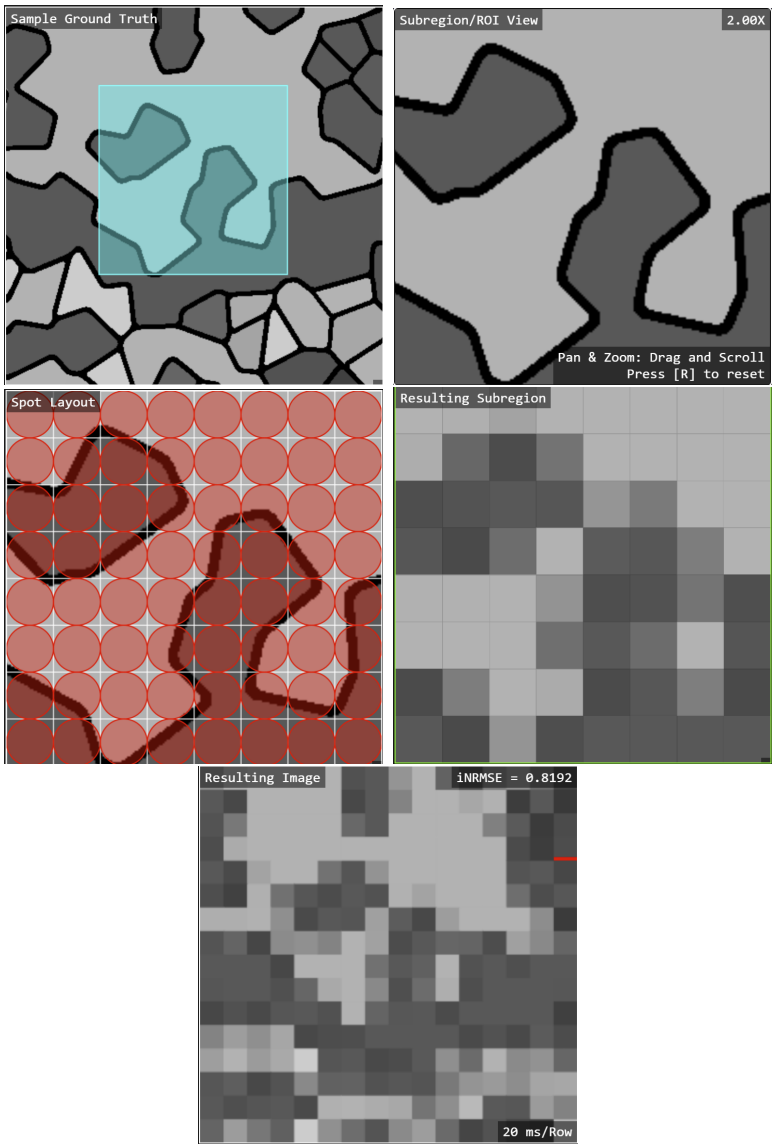

Figure 9: The spot layout, subregion, and resulting image matching the expected test result.

4.3 Image Quality Metric (R7)

This section focuses on testing the image quality metric general cases. Naturally, this is no flawless or foolproof image quality metric. Over 20 different image metrics have been reviewed and compared by Jagalingam and Hegde in a 2015 paper, each with their different strengths and weaknesses [6].

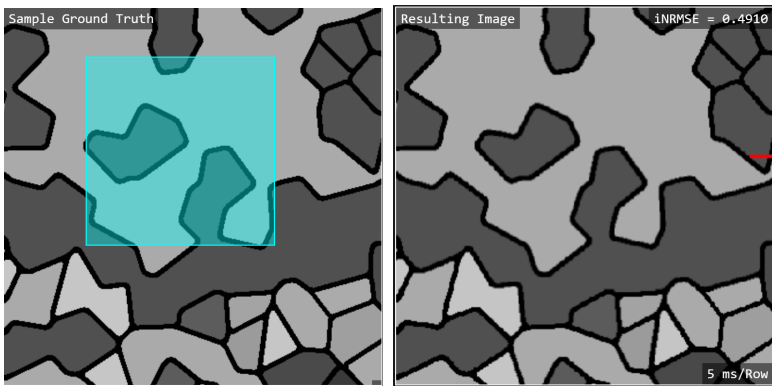

Figure 10: The expected test result: the ground image and resulting image are visually identical.

4.3.1 T10: High metric value (approximate to exact-sampling)

The test was passed as shown in figure 11.

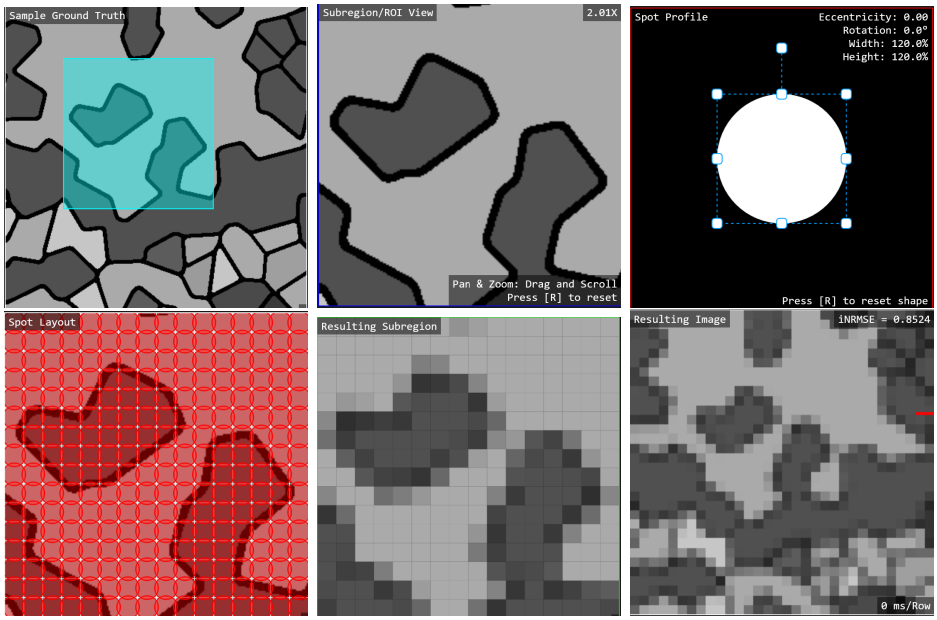

Figure 11: The spot layout, spot profile, subregion, and resulting image with a score equal or greater to the expected test result of a “0.8501” minimum.

4.3.2 T11: Low metric value (under-sampling)

The test was passed as shown in figure 12.

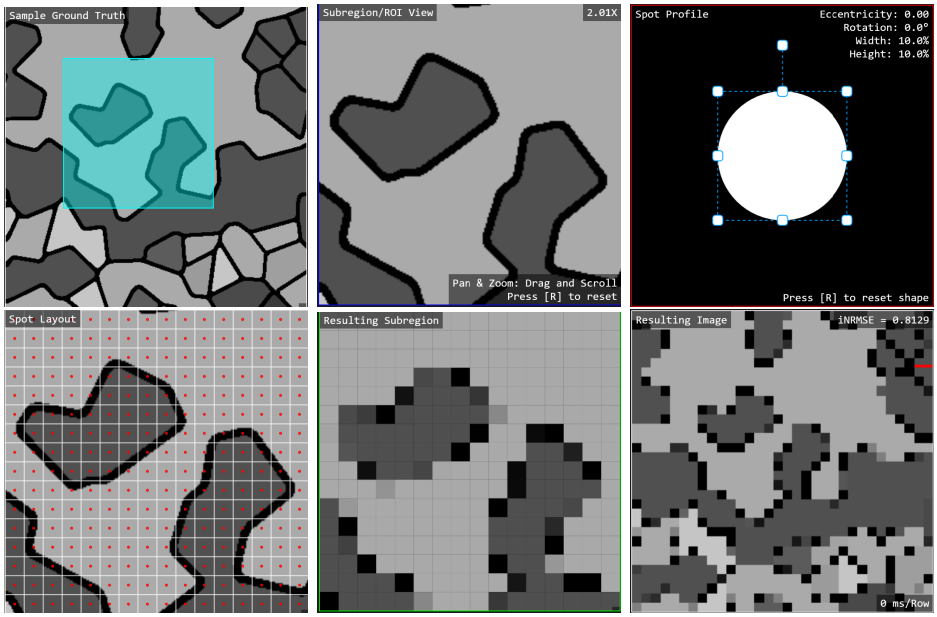

Figure 12: The spot layout, spot profile, subregion, and resulting image with a score less or equal to the expected test result of a “0.8501” maximum.

4.3.3 T12: Low metric value (over-sampling)

The test was passed as shown in figure 14.

4.3.4 T13: Control metric value

The test was passed as shown in figure 14.

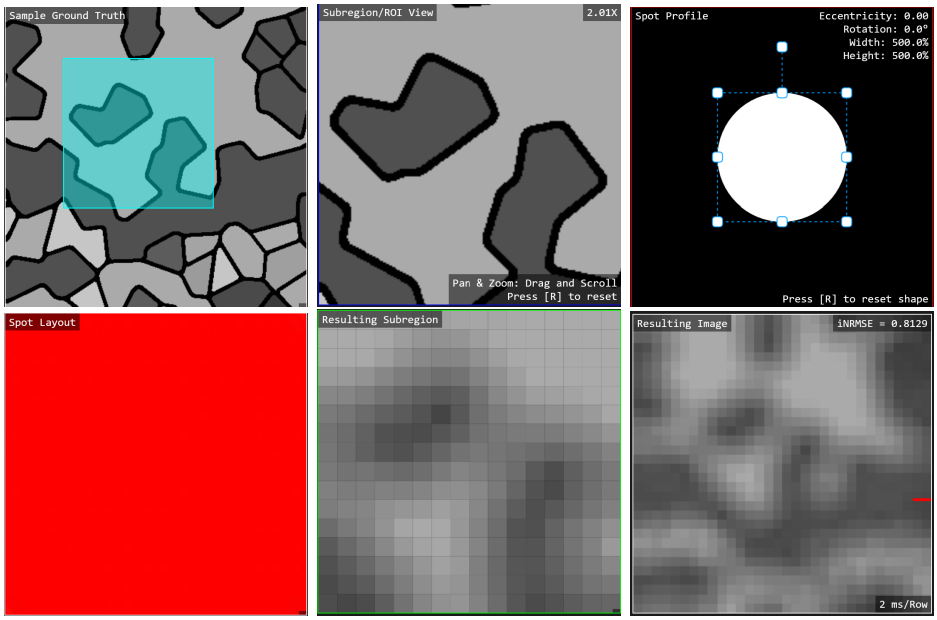

Figure 13: The spot layout, spot profile, subregion, and resulting image with a score less or equal to the expected test result of a “0.8501” maximum.

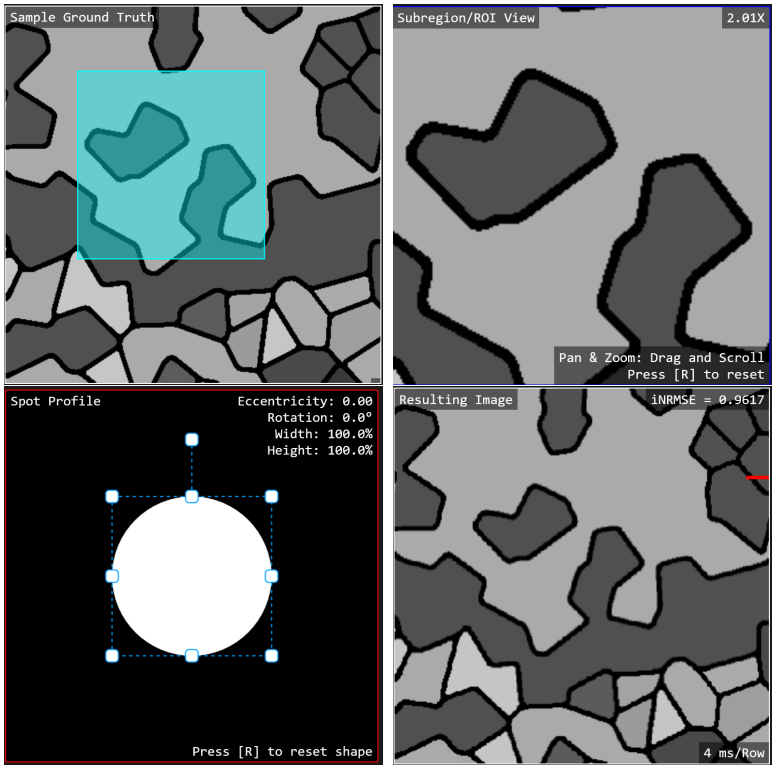

Figure 14: The spot layout, spot profile, subregion, and resulting image with a score greater or equal to the expected test result of a “0.9500” minimum.

5 Nonfunctional Requirements Evaluation

This section focuses on the testing results verifying whether the nonfunctional requirements (as defined in the SRS [3]) are met. Emphasis is put on the *usability* and by extension *portability*: The software should be easy to set up and use without having to worry about any technicalities that could otherwise hinder or completely prohibit non "tech-savvy" individuals from using the software.

5.1 Usability

5.1.1 T14: Usability Survey (NFR2)

The usability survey (see VnV Plan [4]) was not completed. The software was casually reviewed in verbal discussion with the expert consultants (as listed in the VnV Plan [4]) throughout the development of the software. Suggested features and minor issues has been implemented into the software such as: the ability to set a spot size by numeric input, draw the resulting row-by-row for responsiveness, and a display for spot shape eccentricity.

5.2 Accuracy

5.2.1 T15: Image Metric Survey (NFR1)

The image quality survey (see VnV Plan [4]) was not executed, but shall be some time in the future.

5.3 Maintainability

5.3.1 T16: Static code analysis (NFR3)

ESList was used to analyze the JavaScript code, and flake8 to analyze the Python code (only the test code involves Python). ESLint reported 186 errors and 20 warnings. Flake8 reported 10 issues. All of which have been resolved. The code linters now return zero issues.

5.3.2 T17: Code Review (NFR3)

A code review was executed by the author, but should be done by another individual some time in the future. Nearly all code that did not respect the code checklist (from the VnV Plan [4]) was rectified. In a few cases, some code has been marked for change with comments prefixed with `TODO:`. Otherwise, all functions except for a few non-design-dependent utility functions have been documented with JsDoc style

comments. As a result, code documentation has been generated and is published at: https://joedf.github.io/ImgBeamer/jsdocs/index.html

5.4 Portability

5.4.1 T18: Various Platforms and Environments (NFR4)

This test was executed by the author and was also indirectly tested successfully a number of times by the expert consultants [4]. Users simply need to navigate to the following link in any modern web browser to run and test the software: https://joedf.github.io/ImgBeamer/app/index.html

6 Comparison to Existing Implementation

This section is not applicable as an existing implementation for public use was not found or is not available for comparison.

7 Unit Testing

As stated, unit testing was limited to image metrics (as explained in the VnV Plan [4]). Unit tests for the image metrics are available under the `tests/image_metrics` folder.

Some statistics on the unit tests:

- A total of 247 tests (13 metric variants with 19 images comparisons each) were executed in 931 ms for the image metrics implemented in JavaScript.
- A total of 152 tests (8 metric variants with 19 images comparisons each) were executed in 38.3125 s for the image metrics implemented in Python.

7.1 Testing Setup

The following are general instructions (repeated here for convenience to the reader) to run the tests (each in their own subfolder).

- `js-tests/`: for javascript implementations
  - open the `index.html` page (using a local web server as described above) and look in the webconsole.
  - Optionally, an online hosted version exists here.

  - mainly in the webconsole, you can use `run_all(fant)` where if `fant` is true, all the image comparison tests will be run using the "fant-sampled" image [5] as the ground truth. Otherwise (false), it will use the "original" image as the ground truth instead.

- `py-tests/`: for python implementations
  - Install Python v3.10.6 or better (has not been tested on other versions)
  - You'll likely need to run `pip install sewar` **once** to get the required image metrics module/library.
  - run `imgquality.py`

7.2 Results for Functional Requirements

7.2.1 Image Metric Control Testing

All these tests passed within the margin of error.

7.3 Results for Nonfunctional Requirements

7.3.1 Image Metric Control Testing

...

7.3.2 Image Metric Experimental Testing

...

8 Changes Due to Testing

1. The passing value for T13 (see 4.3.4) had to be changed from "0.9800" to '0.9500' since the resulting images look identical. The resulting value of '0.9617' seemed satisfactory and the original expected test value was perhaps too constrained.

2. For similar reasons, the error margin for the unit tests was adjusted from ± 2% to ± 5%.

3. Code is cleaner/clearer and better commentated/documented after running code linting and doing code review. Many unnecessary global variables have been eliminated.

9 Automated Testing

Currently, this is no automated execution of the tests every time there is change. That said, the tests are easy to run (just executing two files) and could be set up with continuous integration some time in the future.

10 Trace to Requirements

11 Trace to Modules

12 Code Coverage Metrics

Code coverage was used as part of this project. However, callgraphs had been generated using Code2Flow to verify the flow application with respect to the design (as per the MG [1] and MIS [2]). An up-to-date but simplified (with limited search depth) callgraph is shown in figure 15. An older (somewhat out-dated) callgraph but with greater detail is shown in figure 16.

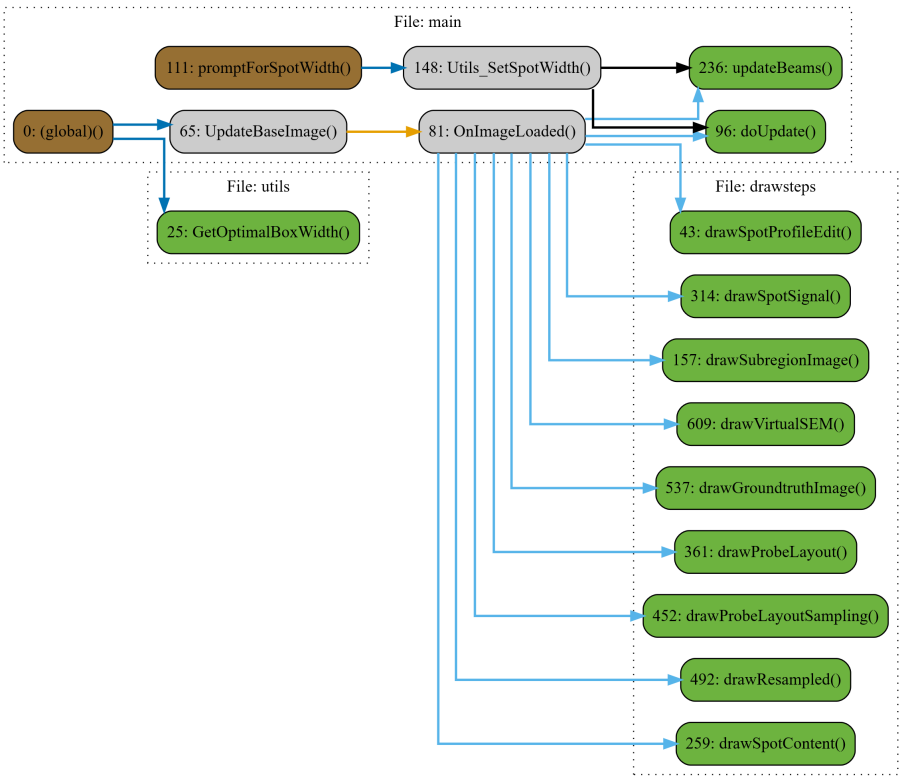

Figure 15: An up-to-date callgraph generated using Code2Flow

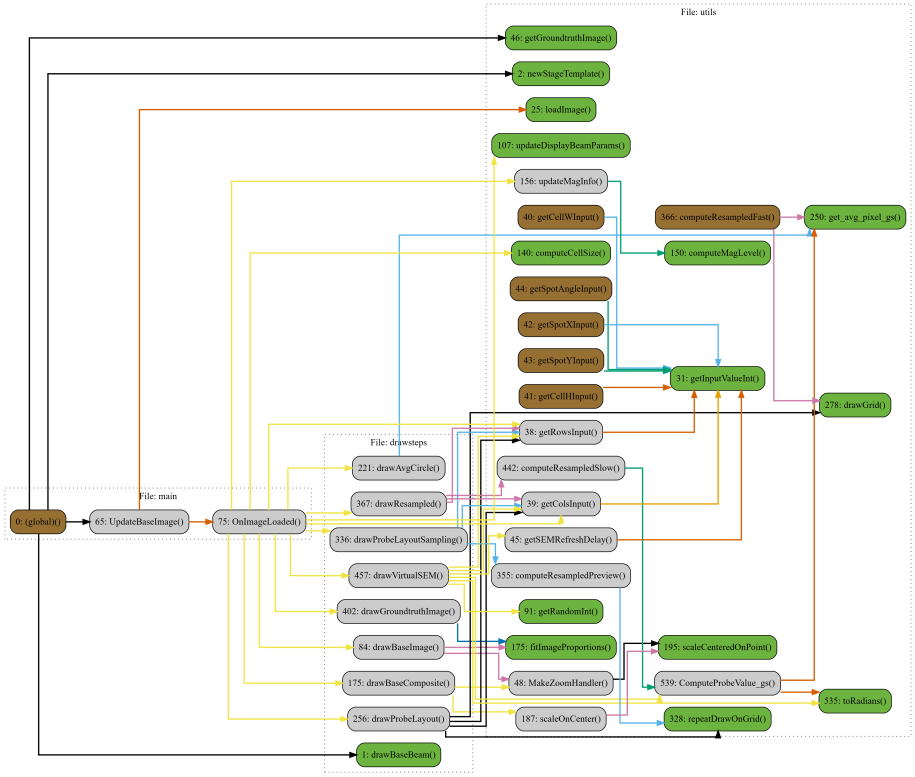

Figure 16: A somewhat out-dated but more detailed callgraph generated using Code2Flow

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. Software requirements specification for ImgBeamer: Scanning electron microscope image formation, 2023. URL https://github.com/joedf/CAS741_w23/blob/main/docs/SRS/SRS.pdf.

[4] J. de Fourestier. Verification and validation plan for ImgBeamer, 2023. URL https://github.com/joedf/CAS741_w23/blob/main/docs/VnVPlan/VnVPlan.pdf.

[5] Karl M. Fant. A Nonaliasing, Real-Time Spatial Transform Technique. *IEEE Computer Graphics and Applications*, 6(1):71–80, January 1986. ISSN 1558-1756. doi: 10.1109/MCG.1986.276613. Conference Name: IEEE Computer Graphics and Applications.

[6] P. Jagalingam and Arkal Vittal Hegde. A review of quality metrics for fused image. *Aquatic Procedia*, 4:133–142, 2015. ISSN 2214-241X. doi: https://doi.org/10.1016/j.aqpro.2015.02.019. URL https://www.sciencedirect.com/science/article/pii/S2214241X15000206.